Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1997), *11*, 59–87. Printed in the USA.

Case-based design: A review and analysis of building design applications

IAN WATSON[1] AND SRINATH PERERA[2]

[1]AI-CBR, Bridgewater Building, University of Salford, Salford, M5 4WT, U.K.
[2]Department of Building Economics, University of Moratuwa, Moratuwa, Sri Lanka

(RECEIVED September 9, 1995; ACCEPTED December 30, 1996)

Abstract

This paper presents a review of CBD and its application to building design in particular. Case-based design is the application of case-based reasoning to the design process. Design maps well to case-based reasoning because designers use parts of previous design solutions in developing new design solutions. This paper identifies problems of case representation, retrieval, adaptation, presentation, and case-based maintenance along with creativity, legal, and ethical issues that need to be addressed by CBD systems. It provides a comprehensive review of CBD systems developed for building design and provides a detailed comparison of the CBD systems reviewed.

Keywords: Case-based Reasoning; Case-based Design; Building; Experience

1. INTRODUCTION

Case-based design (CBD) is the application of case-based reasoning (CBR) to design; thus, solving design problems by adapting solutions that were used to solve previous design problems. Design is an ill-structured domain (Simon, 1973) where knowledge required for problem solving cannot be formalized into a robust model. This makes traditional artificial intelligence (AI) techniques such as rule-based expert systems inadequate for design problem solving. Designers use context-based knowledge and problem-solving skills along with previous design experience to solve design problems. The uniqueness of a design solution often depends on the creative ability of the designer to satisfy constraints within the problem context.

Because experience plays a key role in developing design solutions, a reasoning method which organizes previous experiences as cases to reason—namely, CBR seems well suited to design problem solving. It is known that designers use their experience of design along with combinations and adaptations of previous designs or parts of designs in creating a new design (Akin, 1986). Schmitt et al. (1994) point out that such adaptations and combinations of previous design features in the architectural design of buildings have resulted in many impressive and innovative designs. Oxman and Oxman (1993*a*,*b*) term this *precedent-based design*.

The design of simple buildings, such as houses, provides a classic example of CBD. In such instances, especially in the case of dwellings in large housing estates, an individual house design is an adaptation of a basic house plan suited to its location and the specific requirements of its prospective occupants. Adaptation of the basic plan may involve the addition of some extra facility to the house, such as a conservatory or garage, and commonly involves creating mirror images or simple rotations of the basic plan.

This paper has the following objectives:

- to briefly introduce the techniques of CBR;
- to analyze the *Design Task* to demonstrate the suitability of CBR for design;
- to identify socio-technical issues in CBD; and
- to provide a comprehensive review, analysis, and comparison of CBD systems developed for building design.

2. CASE-BASED REASONING

The origins of the present state-of-the-art in CBR date back to the work of Schank and Abelson (1975, 1977) on dy-

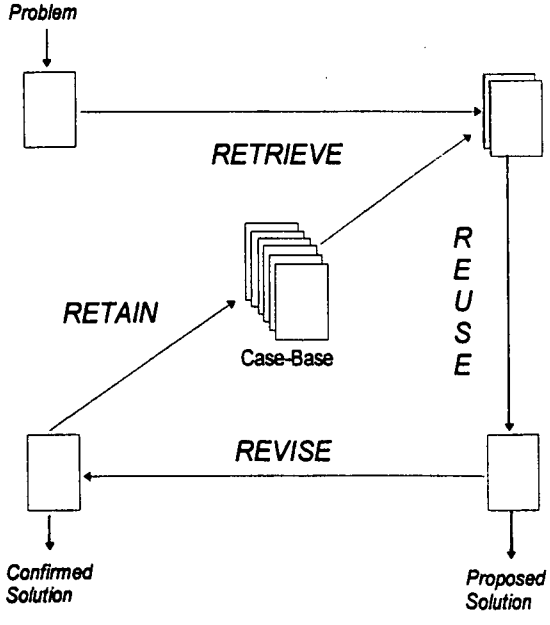
Reprint requests to: Dr. Ian Watson, AI-CBR, Bridgewater Building, University of Salford, Salford, M5 4WT, U.K. E-mail: i.d.watson@surveying.salford.ac.uk.

namic memory and the central role that earlier situations and situation patterns have in problem solving and learning (Slade, 1991; Kolodner, 1993; Aamodt & Plaza, 1994; Watson & Marir, 1994). Case-based reasoning has grown out of psychological models of episodic memory and the technological impetus of AI. Thus, CBR provides a methodology for building intelligent systems and a cognitive model of reasoning.

In the words of Slate (1991):

> Expertise comprises experience. In solving a new problem, we rely on past episodes. We need to remember what plans succeed and what plans fail. We need to know how to modify an old plan to fit a new situation. Case-Based Reasoning is a general paradigm for reasoning from experience. It assumes a memory model for representing, indexing and organising past cases and a process model for retrieving and modifying old cases and assimilating new ones. Case-Based Reasoning provides a scientific cognitive model.

Case-based reasoning is commonly described as a cyclical process as shown in Figure 1 [after Aamodt and Plaza, (1994)]. A CBR system *retrieves* a suitable case from the case library by matching indexes established for the new problem. The information and knowledge in the retrieved case are then *reused* to provide an initial solution to the problem posed. Where the initial solution does not fully satisfy the problem specification, the retrieved case's solution is adapted using domain rules, heuristics, or human intervention; that is, the solution is *revised*. The adapted solution is evaluated to assess the suitability of the new solution. If it provides a sufficiently valuable solution, it may be *retained* and added to the case library.



The next section examines design and assesses how it matches the CBR cycle.

3. DESIGN TASK ANALYSIS

The design of a complex artefact such as a building involves a process with distinguishable stages, each generating more detail in the design (Mackinder & Marvin, 1982; Perera, 1989). In general, any design involves mapping from the design specification to components and building elements, that is, mapping from behavior to structure. This typically involves a search or exploration of the possible subassemblies of constituent components (Fig. 2). Hence, design is a synthesizing task (Chandrasekaran, 1990).

This paper does not discuss design methods in detail, as this is not the objective of the paper. The propose critique modify (PCM) design methodology of Chandrasekaran (1990) is described to indicate the suitability of the design task to CBR. Further discussion and analysis of other design methods can be found in Asimov (1965), Pahl and Beitz (1984), Darke (1984), Dixon (1988), Tomiyama and Yoshikawa (1987), Mostow (1989), Gero (1990), Suh (1990), Flemming et al. (1992), Fey and Vertkin (1993), Arciszewski and Michalski (1994).

3.1. Definition of the design task

A designer is assigned the task of specifying an artefact that delivers some function and satisfies some constraints using a set of design primitives, subject to interrelationships of components. For example, an architect can assume the availability of walls, floors, roofs, doors, and windows as design primitives for a building. Chandrasekaran (1990) defines the design task as follows:

> The design problem is specified by (1) a set of functions (those explicitly stated by the design consumer as well as those implicitly defined by the domain) to be delivered by an artefact and a set of constraints to be satisfied and (2) a technology, that is, a repertoire of components assumed to be available and a vocabulary of relations between components.
>
> The constraints might pertain to the design parameters themselves, the process of making the artefact, or the design process. The solution to the design problem consists of a complete specification of a set of components and their relations that together describe an artefact that delivers the functions and satisfies the constraints.

Hence, design is a recursive process where primitives are adjusted to satisfy the specification for the artefact to obtain the desired function while satisfying conditions of interrelationships between components. The fact that at the

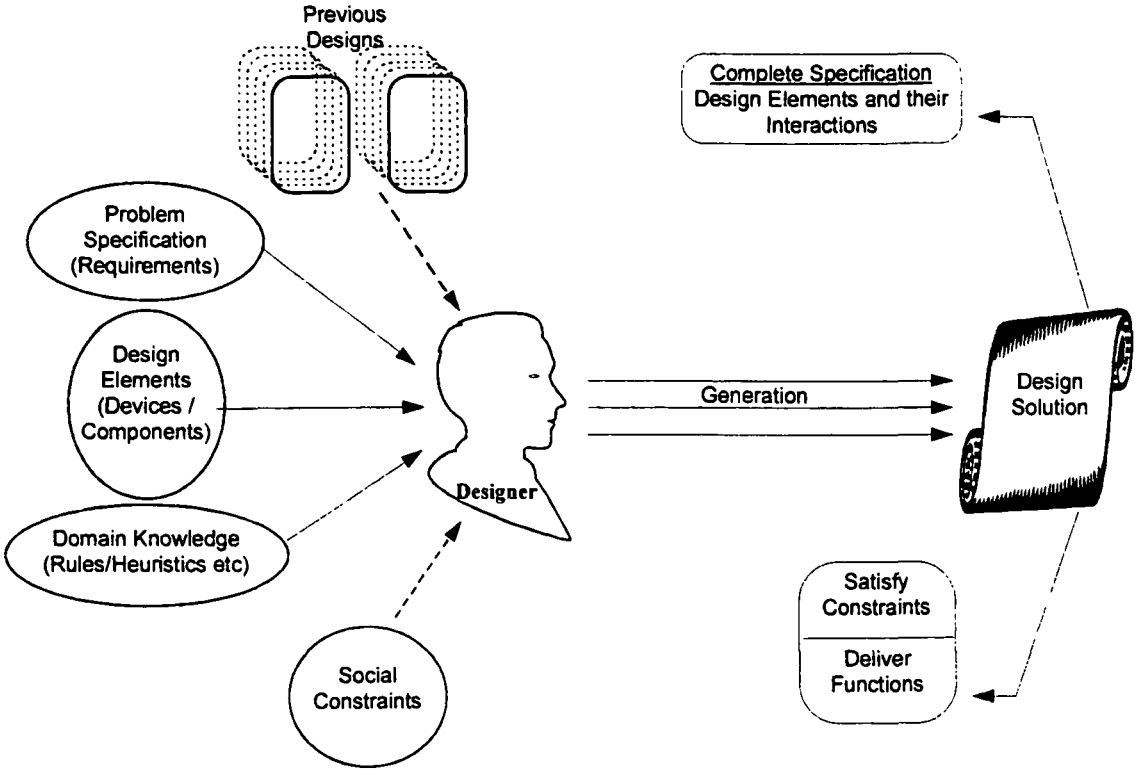


Fig. 2. The design task.

inception of a design the design problem specification is often minimal in terms of functional constraints, contributes to the recursiveness.

3.2. Use of methods in the design process

A method can be described in terms of the operators it uses, the objects it operates on, and any additional knowledge about how to organize operator applications that satisfy the goal (Chandrasekaran, 1990). He further classifies design problem-solving methods into either:

1. a problem space search [after Newell (1980)], or
2. algorithmic solutions.

Algorithmic solutions are more applicable to structured domains. But for ill-structured domains, such as design, a problem space search or a combination of both methods are more relevant. Chandrasekaran calls these methods *Propose–Critique–Modify* (PCM). These have the subtasks of proposing partial or complete design solutions, verifying proposed solutions, critiquing the proposals by identifying causes of failure if any, and modifying proposals to satisfy design goals. Figure 3 provides an overview of the PCM process, each of which is described below.

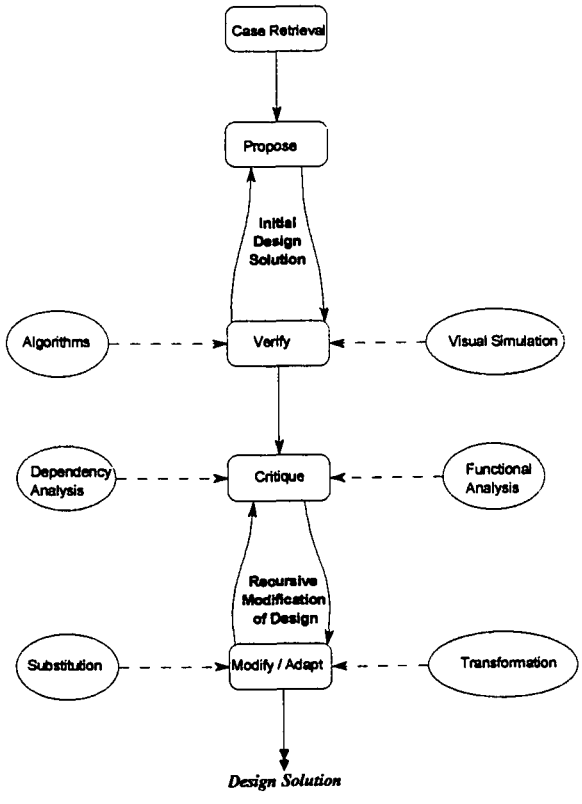


Fig. 3. Overview of PCM method for design.

- **Propose**—involves using domain knowledge to map part or all of the specification to partial or complete design proposals. This involves:

  1. problem decomposition,
  2. retrieval of designs from memory—that is CBR, and
  3. constraint satisfaction and solution composition.

  The first method uses domain knowledge to map subsets of design specifications into a set of smaller design problems. Case-based reasoning retrieves designs from memory to propose a design as a solution or partial solution. Constraint satisfaction uses a variety of quantitative and qualitative optimization and constraint satisfaction techniques.

- **Verification**—is the process of checking that the design proposal satisfies functional and other specifications. It can either be by use of domain-specific algorithms or by visual simulation. Case-based design systems use both verification methods.

- **Critiquing**—is the assessment of the proposed design solution. Here instances of a design's failures may be analyzed. Parts of the design are identified as potentially responsible for unacceptable behavior or constraint violation; that is, mapping from undesirable behavior to parts of the structure responsible for that behavior. Two methods are commonly used for this purpose:

  1. *Dependency Analysis* (Stallman & Sussman, 1977) is used where explicit information is available in the form of knowledge that explicitly relates types of constraints or specification violations of prior design commitments.
  2. *Functional Analysis* (Goel, 1989) identifies violations of behavior or relationships between structure and the intended functions.

  Both of these methods are used for critiquing and the subsequent adaptation of design in CBD systems. It is often difficult to differentiate between the two methods in practical instances of critiquing. However, some form of critiquing is a prerequisite for design adaptation.

- **Modification**—takes information about a failure of a proposed design as its input and then changes the design to get closer to the desired specification. Domain-specific knowledge can be used to guide the adaptation process. The strategies available are numerous and their usage in CBD systems vary considerably.

The design methodology described in this section clearly demonstrates parallels with the CBR cycle, as is shown in Figure 4.

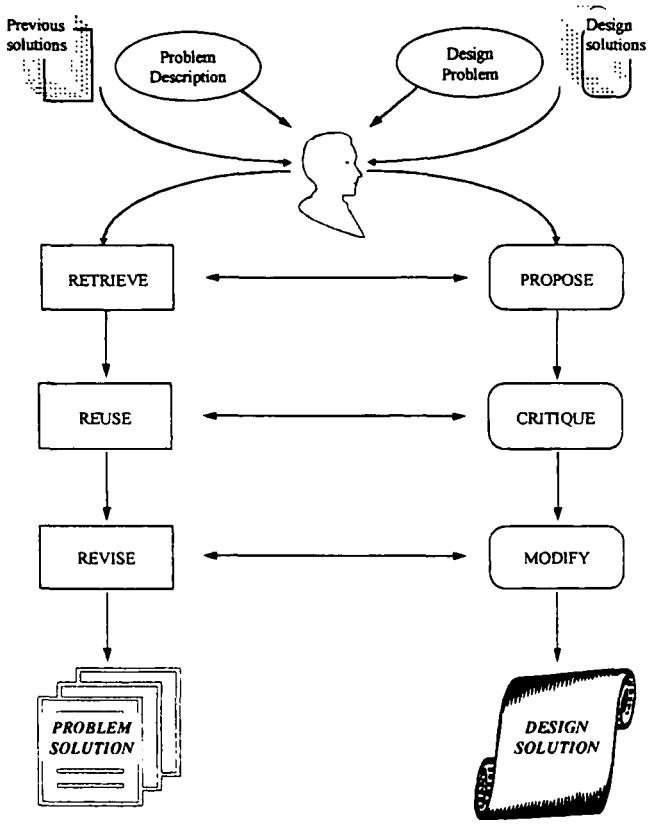


Fig. 4. Mapping the design task to the CBR-cycle.

4. CASE-BASED DESIGN

Case-based reasoning is a useful tool for intelligent system development in a domain where either an explicit model does not exist or one is not yet adequately understood (Kolodner, 1993). Design is one such domain. Simon (1973) describes design as an ill-structured problem, and Maher and Zhang (1993) state that design experience plays an important role. This supports the suitability of CBR for design problem solving. The analysis of the design task using the PCM method of design also clearly supports the suitability of CBR for design.

During the design process, designers reason using previous designs (either parts of designs or whole designs) (Mackinder & Marvin, 1982; Akin, 1988; Schmitt, 1993*a*; Bartsch-Sporl, 1995). In CBD the designer is offered previous solutions to a similar problem indicating how a previous combination of constraints was handled (Schmitt, 1993*a*). This process of using previous designs in the creation of new designs is case-based design. Case-based design can be defined as:

> The process of creating a new design solution by combining and/or adapting previous design solutions.

Maher and Zhang (1993) describe CBD as a hybrid approach, as it uses specific design cases in conjunction with

generalized or compiled knowledge. It provides the designer with at least a starting point if not a complete or comprehensive solution.

The following sections examine how CBR was first used in design and discuss its applicability to design in more detail.

4.1. History of CBD

The earliest CBD systems can be found in work related to AI in design. These were developed as expert systems for design and were aimed at automating routine design tasks mostly at the conceptual design stage in mechanical engineering (Dixon & Simmons, 1983, 1984). PRIDE (Mittal et al., 1985), an expert system for the design of paper handling systems in copiers, used previous designs for routine design tasks and design experience held in a large knowledge base to solve more complex design problems. This work tends to conform with the PCM method and thus conforms to the CBR cycle.

A clearer example of CBD comes from the work related to the subsequent development of STRUPLE by Maher (1987). STRUPLE is an expert system for preliminary structural design in which an intelligent database interface was developed. The database STRUPLE used contained designs of multistorey buildings (this system will be discussed in detail in Section 6).

CYCLOPS (Navinchandra, 1987, 1988, 1991) is widely considered as the first true CBD system. It combines constraint-based solution generation with case-based debugging and repair for landscape designs (this system will also be discussed in detail in Section 6).

The concept of using *design prototypes* as the basis for creative design (Gero, 1987, 1990; Rosenman & Gero, 1993) also relates to CBD and its early developments as a research paradigm. A system based on design prototypes as a generalization of design elements provides a framework for storing design experience incorporating necessary functional, behavioral, and structural information. These concepts have been used in CBD systems such as CADSYN (Maher & Zhang, 1991).

Interestingly, none of these systems used the term CBD. Navinchandra (1987) used the term "*precedent-based design*," while Maher (1987) used the term "*experience-based design*." She used the definition of "*analogical reasoning*" by Carbonell (1986) as the theoretical basis for her work in creating STRUPLE. It was the first workshop on CBR (Kolodner, 1988) that saw the formalization of CBD as a distinct research area. This workshop saw the presentation of many CBD systems from various design disciplines (Pu, 1993), including: JULIA (Hinrichs, 1988), a meal planning system, and CYCLOPS. These were soon followed by many other CBD systems, such as KRITIK (Goel, 1989), ARCHIE (Goel et al., 1991), CADSYN, CAB-Assembly (Pu & Reschberger, 1991), CADET (Navinchandra et al., 1991; Sycara et al., 1992; Sycara & Navinchandra, 1992), DEJAVU (Bardasz & Zeid, 1993). All of these systems address issues of CBR related to design.

4.2. Overview of CBD approaches

From this section onward, in line with the objectives of this paper, attention focuses on building design. Due to the complexity of buildings, design commonly involves many participants thus creating different perspectives of the design including: architectural, structural, and services design plus cost-estimating perspectives. Of these perspectives, architectural design forms the core. Schmitt (1993*a*) defines architectural design as "the art of producing a complete building specification from an incomplete problem description." During the design process, architects reason using previous architectural design cases. During such rememberings, the knowledge of significant design concepts may be derived from past designs to aid the current design solution. This process, of using previous design cases or precedents has also been termed *precedent-based design* (Oxman & Oxman, 1993*a*,*b*).

Domeshek and Kolodner (1993) identified the emergence of two approaches in the use of CBR for design subsequent to the 2nd International Conference on AI in Design, held in 1992 at Carnegie Mellon University Namely:

1. Systems that help designers recall past designs—these use libraries of designs to remind the user of appropriate design solutions. ARCHIE, ARCHIE-II (Domeshek & Kolodner, 1991), MEMORABILIA (Oxman, 1993, Oxman & Oxman, 1993*a*,*b*), CASECAD (Maher & Balachandran, 1994*a*,*b*) are examples of design recall systems.
2. Systems that aim at automating design either fully or partially—these use libraries of previous designs and retrieve appropriate designs for adaptation either by a designer or by the system. CADRE, CADSYN, and NIRMANI (Perera et al., 1995; Perera & Watson, 1995, 1996) are examples of such systems. Design automation systems use varying degrees of adaptation support. Some leave adaptation to the user but provide adaptation verification knowledge. Others help the user adapt the design with modification and verification knowledge, while some provide fully adapted solutions to the designer.

However, Domeshek and Kolodner (1993) and Raphael et al. (1994) point out that both of these approaches raise many of the same issues of representation of design problems and solutions, segmentation of the representations into useful chunks, indexing of the resulting chunks for retrieval at appropriate times, content analysis, and tracking the knowledge requirements throughout the design process.

Schmitt (1993*a*) identifies the following major characteristics of CBD systems:

1. A CBD system does not require a complete domain model but can produce complete and complex designs based on a relatively small knowledge base.
2. Design starts from complete cases, implicitly achieving trade-offs between several constraints.
3. Applying the design history of existing cases can make design problem solving more efficient.
4. Using cases as the source of knowledge allows learning by storing new cases.

These characteristics are inherent to CBR systems. However, the third point leads to an important issue that CBD systems for building design must address. This relates to the fact that designs are context dependent. Building designs are a product of their environment and therefore a solution derived from a past design needs to be contextually compatible. This issue is examined in Section 5.

4.3. Using CBR for design

The design process relies on many types of knowledge, such as design styles, heuristics, and other domain knowledge. Maher and Zhang (1993) consequently describe CBD as a hybrid method because it uses specific design cases in conjunction with generalized knowledge in the form of design rules, or causal models.

Figure 5 depicts a framework for CBD systems that integrates designers' experience in the form of cases with domain-specific and context-based knowledge to solve design problems.

In a CBD system the experience of designers can be captured as design cases stored in a case base. Design cases can be indexed and retrieved to provide solutions to design problems. However, where wholly acceptable solutions cannot be found, domain-specific knowledge-guided adaptation may be required. Domain-specific knowledge may be represented as domain rules or models.

Sycara et al. (1992) identify characteristics of domains where CBR is most applicable, taking into consideration efficiency as a desideratum in knowledge acquisition, system implementation, and maintenance. These are illustrated in Table 1, which maps CBR criteria to building design in a similar way that Sycara et al. (1992) did for mechanical engineering design. It clearly illustrates the usefulness of CBR in building design and also the similarity of CBD approaches to the way buildings are designed.

5. SOCIAL AND TECHNICAL ISSUES IN CBD

In this section we identify several issues that affect the usefulness of CBD systems for building design. Case-based design issues can be categorized as *social* or *technical*. Social issues deal with problems of creativity, ethical, and legal issues related to storing and reusing design cases. Technical issues deal with aspects of case representation, retrieval, presentation, adaptation, combination, and maintenance.

5.1. Social issues

Social issues are important as these directly affect any practical use of CBD systems. The degree of success in address-

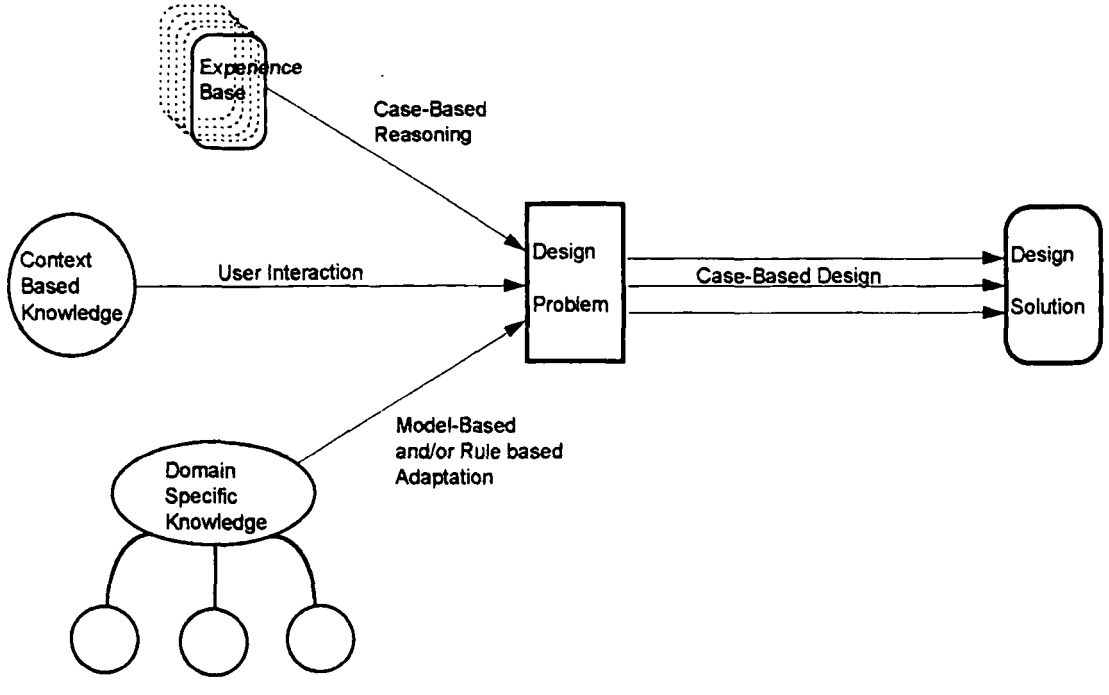


Fig. 5. A framework for CBD systems.

Table 1. *Compatibility of CBR with design criteria*

Criteria	Building Design
An expert knows what he/she means by a case.	A case is a previous building design or part of a building design.
Domain experts draw inferences from comparing a current problem to cases.	Design experts generate designs from prior cases and use analytical models to verify that the generated design meets its specifications.
Experts adapt cases to solve new problems.	Design specifications, simulation, and prototyping results guide adaptation of design cases.
Cases are available in bibliographic sources, in experts' memories, or can be recorded as new solutions are generated.	Design cases are readily available in design catalogues and record books. Companies that keep records of designs they generate try to reuse the designs when similar tasks or problems are encountered.
There are means in the domain to assign an outcome to a case, explain it and deem it a success or a failure.	Simulations, prototypes, and field tests are means by which designs are tested and evaluated. However, building design prototypes are rare. User or expert evaluation of designs and constructed buildings are commonly used.
Cases can be generalized to some extent. Features that make them relevant can be abstracted.	Dissimilar structural configurations can deliver the same behavior. Hence, behavioral descriptions are natural abstractions of design.
Comparisons can be implemented computationally with some level of effectiveness.	Case comparisons and adaptations can be done effectively.
Cases retain currency for relatively long time intervals.	Designs retain currency for long periods of time. For example, the basic design of a door has not changed for several hundred years. Technological innovations may cause design adaptations.
The domain may, or may not, have a strong model.	Despite the existence of physical laws and principles, design is a creative and poorly understood process.
Cases are used in training professionals in the domain.	Building design students are taught design through the use of numerous cases. When an entry-level architect or engineer joins a design office, an important part of his training involves going through the design records of previous projects.

ing these issues, therefore, will directly affect the success of CBD systems in practice.

5.1.1. Creativity

Schmitt (1993*a*) defines creativity as "*the art of causing to exist original ideas or objects.*" Interpretation of creativity varies from a view of creativity as a *mystical activity* to Schank's provocative statement that "*creativity is mechanical*" (Schank, 1986). We consider creativity to be more a social than a technical issue because it is the way in which society perceives a design that will ultimately judge it to be creative or not. Creativity in a design is society's interpretation of the success and novelty of the design. However, CBD systems can incorporate technical features to support or enhance creativity (these will be discussed in detail in Section 5.2).

Maher (1994*a*) states that new designs and corresponding design knowledge are based on existing designs and design knowledge. If invention or discovery takes place by combining ideas, we can view creative design as a juxtaposition of designs or design styles that have worked well in the past (Goldberg, 1989). Schmitt quotes Brock's (1992) definition of creativity as "*giving a new order to existing components.*" Gero and Maher (1992) identify routine, innovative and creative design, and offer graphical explanations of the creative solutions that may be formed. Rosenman and Gero (1993) explain creative design as dealing with the formulation of new structures. That is, the creation of new vocabulary elements or new configurations of existing vocabulary elements in response to either existing or new functional requirements.

In examining these and other definitions, it can be concluded that design creativity does involve some elements of the reuse of past experience in designing similar artefacts and reorganizing elements in a novel and useful manner (Hayes, 1981; Oxman, 1993). Thus, creativity can be said to build on knowledge and experience.

Table 2 illustrates how CBD supports creativity in building design. It has been compiled from views expressed by Domeshek and Kolodner (1993), Maher and Zhang (1993), Oxman (1993), Pu (1993), Schmitt (1993*a,b*), and Maher (1994*a*). These clearly indicate that CBD systems can be used in a manner that would enable the human designer to improve designs by enhancing creativity. Case-based design systems can provide useful remindings of previous instances of a particular form of design. Case-based design can also provide the designer with useful insights as to the way in which certain sets of design components were combined to provide the designed performance of the artefact.

From a different perspective, it can also be argued that CBD systems could negate creativity. This could happen where a blind use of previous designs is encouraged. Certainly, use of previous designs could force the designer in a particular direction. But we believe it is the way in which system designers envisage the use of previous designs in the creation of new designs that govern whether a CBD system would enhance or negate creativity in design.

Table 2. *Satisfying the aspects of creativity in CBD*

Aspect of Creativity	How CBD Satisfies Creativity
Knowledge of a set of precedents. Past cases of building designs and components or elements.	Indexed case library or an object-oriented or relational database storing previous design cases.
Creativity is a function of the designer's ability to explain the precedents and their reasons for being in the case base.	Perform inferences or explanation by identifying similarities and differences in cases.
Creativity relies on heuristics to find applicable solutions.	Heuristic search and inductive reasoning to select an appropriate case from a case library.
Creativity builds on the capacity of the designer or an external critic to ask harsh, uncomfortable, or seemingly unrelated questions.	This is related to inductive reasoning and is almost the inverse of deductive inference.
Creativity is a function of idiosyncratic experience of individual designers.	Use of multiple case libraries containing the design experience of many designers.

5.1.2. Ethical and legal issues

The use of information technology in industry has raised many ethical issues. Huff and Martin (1995) recently identified AI as one of the key areas that needs a greater degree of consideration in terms of ethical issues. For example, in a CBD system designs could be by different designers. Using many design cases from different designers may benefit the CBD process by providing a richer knowledge source. This clearly attracts not only ethical issues but also legal issues. Is it ethically correct to combine or modify designs of other designers? This may also cause legal issues relating to copyright.

One solution to this problem could be to limit the designs in a case base to those of one particular designer or design organization. This may avoid legal claims but may result in an insufficient number of cases to populate the case base.

In terms of design education, Schmitt (1993*a*) points out that the worst case scenario is that reasoning with cases might cause plagiarism and the inappropriate combination of elements or features. As a solution he proposes CBD systems should be used by upper level design students rather than beginners, for whom he prescribes bottom-up design methods, such as grammar-based design.

A more detailed discussion of ethical and legal issues exceeds the scope of this paper. We recognize this as one of the major areas to be investigated by CBD system developers. The fact that the *Communications of the ACM* has devoted one full issue to the subject of ethical issues in the use of information technology confirms our view and indicates that it is receiving some attention (*Communications of the ACM*, 1995).

5.2. Technical issues

In this section we examine in detail the issues that concern CBD system development and draw examples from CBD systems found in the literature.

Sycara (1992) identifies four challenges for CBD systems.

1. Relating behavior and structure. The design process involves the transformation of largely ill-defined functional descriptions for an artefact into detailed physical description that satisfies the functional. This requires reasoning at different levels of abstraction ranging from physical to functional. Thus, case representation should support vocabularies that express and capture relationships between function, behavior and structure of artefacts (Sycara & Navinchandra, 1989).
2. Designs often represent a tight integration of numerous components or elements. Thus, indexing schemes should support interaction and integration of elements.
3. Because functional description of the design object is often under-specified, the designer is dynamically generating subgoals and filling information gaps. Thus, an indexing system should generate indexes dynamically for generated subgoals.
4. Case-based design systems should allow access to individual element or component designs separately, but at the same time the system should consider element interactions and relationships, thus maintaining the overall performance of the ultimate artefact.

These points form a broad performance specification for CBD systems. They indicate the level of complexity in design that is needed to be accommodated by CBD systems. To analyze these and other issues related to CBD in a structured manner, we define the issues to be: case representation, case retrieval, case presentation, case adaptation, and case-base maintenance. These central issues of CBD are in fact the same as those of CBR in general. The following subsections examine each of these issues in more depth.

5.2.1. Case representation

Representation of cases is fundamental to a CBD system. Representation forms the core of a CBD system on which all other issues depend upon. Maher (1994*a*) identifies three main issues in case representation:

1. the content of design cases,
2. the case memory organization, and
3. the presentation of design cases to the user.

Of these issues, the first two will be examined here while case presentation is discussed in Section 5.2.3.

To identify the content of a design case, it is important to find out how design and design requirements are represented in practice. Requirements for a building are explained in terms of client briefing documents as text and as verbal communications. The final design solutions are represented as drawings, specification documents, Bills of Quantities, or cost reports and other performance specifications. A case is the final complex result of a successful design process. It does not necessarily reveal causal relations between design decisions, but it may lead to the discovery of such relations (Schmitt, 1993*b*).

Raphael et al. (1994) state that design cases should contain the following:

- a problem specification for the design including requirements,
- a final design solution,
- the rationale behind the design solution,
- an evaluation of the design, and
- histories of successful as well as unsuccessful designs,

Of these, the information included in a design case is often determined by the problem or subproblem the reasoner is solving. Our experience of developing CBD and estimating systems at Salford University (NIRMANI and CBRefurb) (Marir & Wilson, 1995*a,b*) and that of others (Hunt & Miles, 1995; Lehane & Moor, 1996) indicates the need for knowledge elicitation to identify useful features of cases. CADET used published data while CADSYN used the developers' own knowledge along with consultation with designers to identify useful case features. This task is complicated and difficult because of the complex nature of building designs, the interaction and interdependencies of elements, the context dependency of design cases and above all the vast amount of information pertaining to a particular building design.

The content of a design case can be represented in many ways. Maher et al. (1995) generalizes these approaches into: attribute-value pairs, text, object-oriented representations, graphs, multimedia representations, and hierarchy-based representations. Most CBD systems use one of these representation methods or variations or combinations of them.

Design cases need to be stored in memory. Two common methods of organizing cases in memory can be found.

1. **Flat**—cases are stored as records of key features (attribute-value pairs) describing the content. Similarity assessment would be on features and their values.
2. **Hierarchical**—cases are clustered into groups according to some features (identified by domain knowledge) and classified in a hierarchy, usually from abstract features to more specific features. Similarly assessment may be on features and their values but can also compare structural similarities between hierarchies.

CYCLOPS uses a flat representation of cases as *unexplained observations, explained observations*, and *explained solutions* organized as attribute-value pairs. STRUPLE also uses a flat representation of cases stored in a relational database and structured into information categories such as: *general information, geometric information*, and *load information*, represented as attribute-value pairs. ARCHIE uses a frame-based attribute-value pair representation that incorporates multimedia features. CADSYN and NIRMANI use a hierarchical case representation that decomposes the design problem into subproblems in the form of super cases and different levels of subcases.

More complex content representations and memory organizations can be found in other engineering design domains such as mechanical engineering. CADET is a CBD system for the synthesis of mechanical devices from cases. It stores cases as acyclic influence graphs, representing the behavior of devices. These are then normalized into a relational database. KRITIK is another CBD system for the design of mechanical devices and electrical circuits. It uses a behavioral component-substance model explicitly representing the structure, intended functions, and internal causal behaviors of devices. Cases are stored in memory as a flat representation with one layer of indexes.

Case representation schema and the memory organization of CBD systems facilitate the efficient and effective retrieval of cases for synthesis and/or justification. The factors to consider here are the ultimate use of the CBD system, its flexibility, and efficiency. Flexibility in storage and retrieval means that the contents of design case memory can shift when new technologies or design styles are being used. Efficiency leads to better performance, especially where case bases become large. The issues of case indexing and retrieval are discussed in the following subsection.

5.2.2. *Case retrieval*

Retrieval is a critical part of CBD systems. This raises issues of design case indexing, retrieval, and selection. Case indexing is closely associated with issues of case content and memory organization. Indexes refer to pointers or labels to cases or case features in the case memory. As in the index to a book, they provide quick access to a design case in memory. Identifying the features to label in cases (the indexing vocabulary) is a crucial aspect of indexing. Indexes should be:

1. Predictive—they should identify features of a case that were responsible for the design solution.

2. Abstract—they should describe the design problem in general and the context within which it was carried out.
3. Concrete—they should describe the specific design problem or subproblem.
4. Useful—they should describe the situations in which a case is useful or features useful for the respective users of a specific design domain (Kolodner, 1993; Maher et al., 1995).

Case-based design systems use various indexing schemes varying from: simple check-list based indexing, as in MEDIATOR (Simpson, 1985), to relationship-based indexing schemes, as in CADSYN, and explanation-based techniques (Barletta & Mark, 1988; Simoudis et al., 1993).

One of the most common techniques used to retrieve cases is the establishment of a similarity metric. This could be on keywords where a text-based representation is used, or attribute values where a feature-based or object-oriented representation is used. Maher and Zhang (1993) point out that retrieval with a fixed similarity metric is a limitation in most current systems. Because the importance of a feature depends on its context, a fixed feature similarity metric would be of less use. However, there are certain simple steps that can be used to achieve a considerable degree of context dependency. For example, where nearest neighbor algorithms are used, the adjustment of the weight of the retrieval criteria can achieve context sensitivity by making some retrieval criteria more important than others in different contexts.

In building, designs evolve from an abstract specification to a detailed one. Therefore, CBD systems should help this process by providing a design exploration space within the system. This could be accommodated with a flexible indexing scheme. Maher et al. (1995) describe three ways to achieve flexibility in indexing design cases:

1. Any set of features in a case description can be used to search case memory.
2. A feature-based, object-based, or graph-based indexing scheme can be used to search case memory.
3. Indexes to case memory can be determined dynamically.

Case indexing is often associated with retrieval strategies. Maher et al. (1995) identify three generic types: list-checking, concept refinement (where a hierarchical indexing tree is used), and associative recall (which uses a relationship-based indexing scheme). Associative recall is more suitable where the design problem is ill-formed such as is typical in building design.

Case-based design systems tend to use varying forms of domain knowledge to assist the retrieval process. STRUPLE uses a similarity metric to compare significant common aspects of a matched case to the current case. CYCLOPS uses causal models, whereas ARCHIE uses qualitative models for retrieval of cases. ARCHIE-II uses descriptive indexes for flexible retrieval of cases and relationship indexes for goal-directed retrievals. CADSYN creates indexes for a whole design case and a feature-based index for the subcases and then uses a hierarchical search algorithm to search the case hierarchies.

NIRMANI uses three feature-based indexing methods to dynamically create indexes for retrieval of cases from its hierarchical case structure. Both CADSYN and NIRMANI support the development of the design problem through the use of an index revision retrieval strategy that provides iterative retrieval of cases.

The end result of a retrieval process is a set of similar and potentially useful cases. Thus, CBD systems provide various ways in which a case can be ranked and selected. The simplest is a count of matching features. The most common is to select a set of indexing features and attach importance values to these features. These importance values can be provided by the user, or can be heuristically, statistically, or analytically derived. In CBD, context-dependent matching is important because building designs are highly context dependent. Retrieving cases out of context would require a greater degree of adaptation (Maher et al., 1995). Hence, some CBD systems use domain-specific knowledge to retrieve and select cases that are of similar context.

CASECAD uses the number of matching features to calculate a match score for ranking retrieved cases. ARCHIE uses weighted feature scores for calculating the total match as per a typical nearest-neighbor algorithm. CADET uses an influence graph similarity assessment to derive a best match based on lowest cost, weight, and ease of synthesis. NIRMANI uses weighted feature scores supported by its indexing methods to enforce contextual compatibility. ADA, an architectural design assistant (Hunt & Miles, 1995), uses a mixture of direct matching and weighted hierarchical matching in ranking retrieval cases. SEED converts its object-based similarity assessment to a match score. It uses an attribute and class classification to discriminate and narrow the search space.

5.2.3. Case presentation

Presenting design cases to the user is one of the aspects many CBD system developers have inadequately addressed. For example, Domeshek et al. (1994) admit that ARCHIE lacks user friendliness and is more useful to a knowledge engineer than an architect. The interface of ARCHIE is too "*painful*" to use by anyone other than a knowledge engineer.

Designers understand and express design problems graphically. Thus, visualization of the design case is of utmost importance in the design task. Visualization enables the designer to understand the underlying design concepts, functionality, appearance, and similar aspects of a design. The importance of case presentation increases if the system is to be an interactive tool (Maher, 1994*a*). To improve the as-

pects of design visualization, a multimedia approach for case presentation can be incorporated in CBD systems (Maher & Balachandran, 1994*a,b*). In addition to textual descriptions or attribute-value pairs, video and photographic images, CAD drawings, charts, audio explanations, and virtual reality simulations can be used to present design cases to the user.

Cases should be presented so as to:

1. Help the designer easily comprehend the design. For example, an architect is more familiar with design sketches, drawings, and models of buildings than with textual descriptions.
2. Provide alternative presentations of the same design so as to enhance the understanding of the design or design subproblem, for example, the use of virtual reality to simulate a design in addition to CAD drawings.

Multimedia presentation of design cases has many benefits. They can:

- convey contextual information surrounding a building design;
- explain and elaborate certain features of a design case;
- simulate behavior through the use of virtual reality or 3D CAD;
- describe problems and constraints associated with certain design features;
- provide pointers to useful documentation, such as evaluations of designs, cost reports, catalogues of materials, components, or equipment;
- convey the design rationale; and
- convey feedback on design solutions.

The benefits of multimedia presentation of cases is evident from the increased use of multimedia in recent CBD systems. ARCHIE-II, CASECAD, and ADA use 2D and 3D CAD drawings, text, and attribute-value pairs, while NIRMANI also uses video, bitmaps, and even pointers to web sites.

5.2.4. Case adaptation and combination

Selecting a case after a retrieval process as a potential solution to a design problem has a different meaning in CBD depending on the intended use of the system. In design aiding systems (e.g., ARCHIE-II and MEMORABILIA), it is a reminder of a situation in which a similar design problem was solved. Thus, these systems point to potential solutions and allow the user to browse solutions, but they do not directly reuse potential solutions to provide a new solution. Design automating systems (e.g., CADRE, CADSYN, and SEED) use a design solution as the basis for providing a new solution to a new design situation. Such systems give a much broader meaning to a selected case.

A selected design case provides a design solution in a particular design context as in Figure 6. In most instances in building design, the design context or the environment and the situation in which the design solution was created are unique. This means a potential design solution (a selected case) must be adapted to conform to the current context. Adaptation in CBD can be defined as the process of modifying a selected case's design solution and making it conform to the new design context.

Design adaptation can be carried out in several ways:

- human intervention—where a designer modifies the design;
- knowledge-based adaptation—where domain-specific or domain-independent knowledge is used to adapt or modify a design;
- case-combination—where design cases are combined to provide new design solutions; and
- combinations of the above approaches.

Kolodner (1993) broadly classifies knowledge-based adaptation methods into four categories:

1. *Substitution Methods*: where selected cases feature values are substituted to provide a new solution.
2. *Transformation Methods*: where rules, procedures, or models are used to transform a selected case into a new solution.
3. *Special Purpose Methods*: where heuristics are used to carry out domain-specific and structure-modifying adaptations.
4. *Derivational Replay*: where the methods or procedures that were used to generate the selected case are reused to generate a new solution.

A survey of adaptation in CBR systems (Hanney et al., 1995) identifies design as having the heaviest adaptation requirements above all other applications of CBR. Different approaches to adaptation have been used by CBD systems. For example, in CADSYN a constraint satisfaction approach is used, whereas CADRE uses dimensional and to-

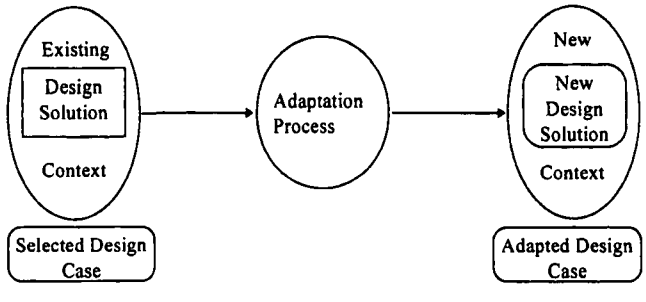


Fig. 6. Design case adaptation.

pological adaptation based on production rules and shape grammars.

Whatever adaptation method is used, extensive domain-specific knowledge is required (Raphael et al., 1994). A case is selected in the first instance because it matches enough of the problem to point to a prospective solution. But adaptation is a process that changes a once satisfactory design. Therefore, the more radical the adaptation, the greater the danger of losing the quality of the original design (Schmitt, 1993*a*). Hence, possible cases should be retrieved so as to minimize adaptation. This is called adaptation-guided retrieval (Smyth & Keane, 1996).

Adaptation knowledge can be formulated in two ways, as illustrated in Figure 7.

1. By categorizing the case as instances of prototypical designs that can simply be reinstantiated. Adaptation knowledge can then be stored as case category specific knowledge. However, this raises the question of why cases are required at all, as prototypes could fulfill the same function to a great extent.
2. By providing specific adaptation knowledge which modifies aspects of cases instead of reinstantiating them (Hua & Faltings, 1993).

Adaptation knowledge can be stored as generalized knowledge in a knowledge base, it can be stored with the cases or it can be supplied by the user at run-time. Any combination of these three methods can also be used.

The concept of prototypical cases and relevant adaptation knowledge is more amenable to CBD systems in domains such as mechanical systems design [e.g., the Motor Design System of Tanaka et al. (1992)]. However, in domains such as building design where prototypical designs are harder to identify, the use of specific adaptation knowledge is more relevant. This is because there could be numerous differences between individual design cases of a particular design prototype and the adaptation knowledge required for all these differences cannot practically be generalized or classified. In CBD systems for building design, varying approaches for storing adaptation knowledge have been adopted. For example, CADRE stores specific adaptation knowledge with cases whereas CADSYN uses a separate generalized knowledge base.

In an attempt to formalize CBD adaptation Maher et al. (1995) identified three components in adaptation knowledge. They describe adaptation knowledge as having modification knowledge and verification knowledge along with adaptation operators that perform the task of adaptation. The strategies used for modification and evaluation of a design case rely on the underlying problem-solving processes. For example, CADSYN uses constraint satisfaction, KRITIK uses model-based reasoning, CADRE uses rule-based reasoning SEED, CAB-Assembly uses heuristic reasoning, and

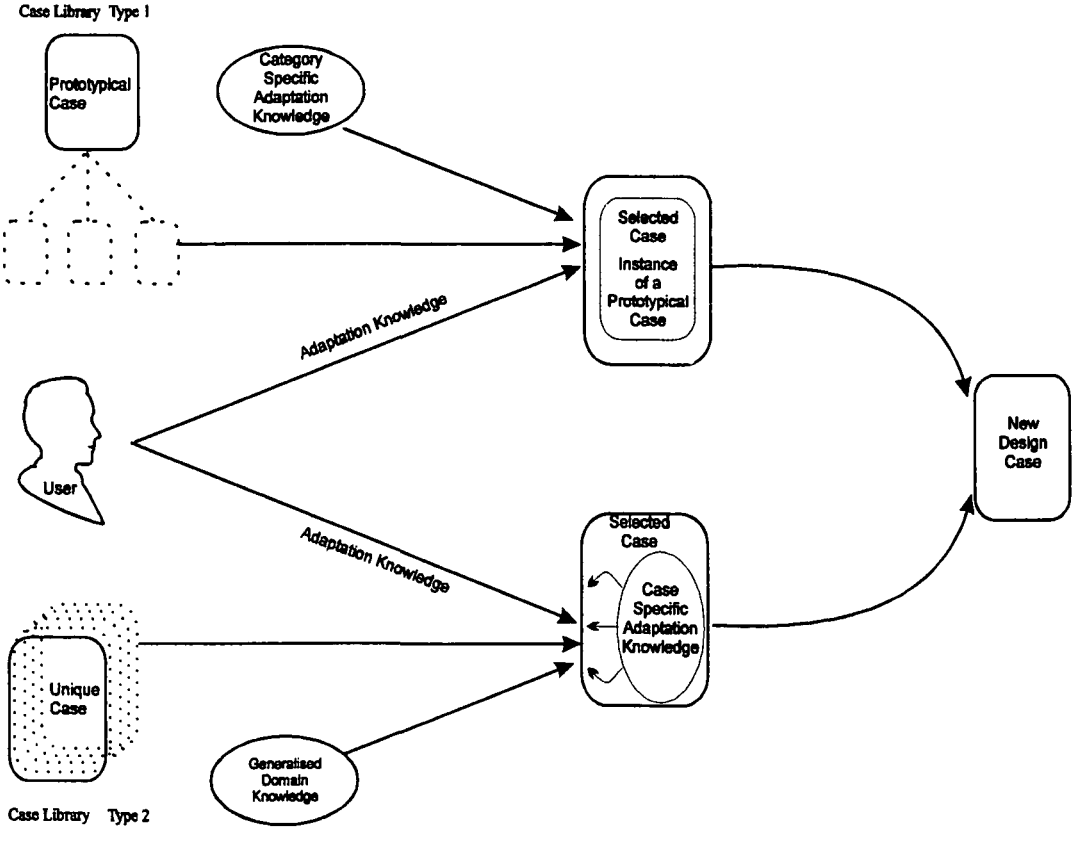


Fig. 7. Methods of organizing adaptation knowledge.

CADET uses qualitative reasoning for synthesis of parts of cases.

CADRE uses design constraints as verification knowledge, production rules, and shape grammars as modification knowledge, and dimensional adaptation as adaptation operators. KRITIK, however, integrates modification and verification knowledge as function, structures and behaviors; while functional mappings between current and previous contexts form the structural operations. NIRMANI uses a rule base of verification knowledge to check for the satisfaction of design constraints. NIRMANI also used adaptation methods and user interaction as adaptation knowledge.

Case adaptation can be manual or automatic or a mixture of both. As pointed out by Raphael et al. (1994), automatic adaptation has many limitations. It is extremely difficult to adapt a design without any violation of interrelationships of design components. This problem is increased because these interrelationships are context based. Schmitt (1993*a*) shows that adaptation occurs at many levels. He further states that where adaptation fails to generate a satisfactory solution, the next alternative is *case combination*. Here, features from one or more cases will be brought in for adaptation of the selected case. Case combination has a sound heritage in architecture. For example, colonial architectural designs in South America, Asia and the Pacific, relied heavily on combining different design cases. Shih (1994) has managed to combine floor plans to create new designs, satisfying new problem specifications. Thus, there has been some success achieved in combination of cases in building design. Much greater levels of success in combining cases have been achieved in other design disciplines, especially in mechanical engineering (Tanaka et al., 1992; Roderman & Tsatsoulis, 1993).

An evolutionary adaptation method (Hunt, 1995) that integrated case combination (cross-over: which substitutes selected case features from other cases) and knowledge-based adaptation (mutation: which modifies case features using domain knowledge) has been successfully experimented with in structural engineering design.

A possible problem in case combination is taking a design element out of context. In certain instances where the design context differs a solution to an identical design problem may vary. For example, the location and positioning of the elevator plant room in a multistorey building is affected by building regulations, the number of storeys, the type of elevator to be installed, and aesthetic requirements. In one instance the plant room may be located on the roof top and in another instance, in the basement. Thus, decisions may be highly context dependent.

However, such problems can be overcome by storing information related to the context with the case. This refers back to the problem of determining the extent of information to be contained in a case. Methods of organizing the contextual information stored with each case are an important aspect for consideration. The *design-perspective* representation of case memory used in NIRMANI stores case information in the perspectives of the various design experts involved in the design process. Context based information is stored within the respective design perspective giving the view of the context applicable to the design element in question.

5.2.5. Case-base maintenance

Case-based design systems complete the CBR cycle by including new design solutions in the case-base. Most CBD systems are initially developed from paper-based or computer-based design cases in the form of documents, drawings, and databases. The question to answer in case collection is "which cases to include?" Kolodner (1993) provides three general principles for this, which are considered in the context of design.

1. Design cases should cover the range of design tasks the system will be required to perform.
2. For these tasks include well-known design solutions and well-known mistakes (design faults, failures, or problems). Successful solutions provide the basis for new solutions whereas unsuccessful solutions avoid the repetition of mistakes by pointing to potential pitfalls. This helps achieve best practice in design.
3. Complete the coverage of design cases by finding what is missing by using the case base.

Schmitt (1993*b*) identifies the following criteria as factors to consider in deciding which cases to include in a case library.

- **Architectural quality**—The building must earn the respect of the professional community as well as the acceptance of its users. Although this is a subjective criterion, it is important.
- **Timelessness**—The design should be a product of its time, but not be merely fashionable. However, where case combination is used design features of an outdated design may be reused in a new design creating a blend of features that may induce creativity.
- **Environmental responsiveness**—The building must offer an appropriate answer to the environmental conditions of the site.
- **Contextual responsiveness**—The building must make a clear statement regarding its position within a context by either adapting to the context or proposing a bold new beginning.
- **Functional quality**—The building must fulfill all functional requirements and in addition offer possibilities for future adaptation.
- **Structural stability**—The design must offer structural safety and also comply with special local ordinances regarding special conditions, such as earthquakes or tornadoes.

However, these criteria may not be fulfilled by many buildings. Thus, a trade-off of a few criteria may be considered in making the final decision. This raises the issue of whether case acquisition should be manual or automatic. This will largely be determined by the individual application environment. If cases are created during the design process, routines can be provided for automated case acquisition. However, this may lead to problems if case quality is not evaluated by using domain-specific knowledge. Another way to acquire cases is from existing databases. NIRMANI uses existing databases to acquire its cases.

Once a CBD system is developed and used for design, it continually creates new design solutions to new design problems. If the system automatically adds the new solutions to the case base it will grow with use. This requires case bases to be maintained. The criteria detailed above to judge the worthiness of a case for inclusion could form a basis for pruning an ever growing case base. Another factor to consider in maintenance of a case base is the frequency of use of the cases themselves. The potential danger of using this as criterion for pruning a case base is that this may limit the capabilities of the CBD system and makes it more specialized in certain type of design. On the other hand, if one considers human designers, one will easily find specialist designers of industrial buildings, offices, hotels, prisons, etc.

Such aspects need to be considered by practical CBD developers even though they do not arise initially at the inception of a system. Unfortunately, most CBD systems in the literature rarely discuss issues of maintenance or management of case bases. This may be due to the fact that the systems are research prototypes and not fielded design systems requiring maintenance of large case bases. However, some reference to case-base maintenance issues can be found in the literature.

CLAVIER, one of the most successful case-based systems in industrial use, classifies its new solutions as successful and unsuccessful using statistical data and expert evaluation (Hennessy & Hinkle, 1992). They do not eliminate the unsuccessful cases, but store them to identify solutions in the future with a potential for failure. SEED leaves the control of the case base to the user or designers. Designers are allowed the option of adding a case to the case base. It considers the growth of the case base as side-effect of a design firms' normal design activities.

CASECAD uses a similar approach. It contains a module for case-base management. One of its functions is to allow the user to create and modify cases. It allows users to add new design solutions to the case base. A case-base *cleansing* mechanism is suggested by Lehane and Moore (1996) and Moore and Lehane (1996). They use the frequency of retrieval as the criterion for pruning a case base. They argue growth of a case base does not deepen or broaden the experience of the library but increases retrieval cases due to increased search space.

NIRMANI also provides a similar approach based on the frequency of usage of cases for case adaptation. A case in NIRMANI contains an attribute that stores a list of cases used in the generation of its solution. This lets us identify cases that are used in the generation of new solutions. However, it does not provide for the automatic pruning of the case base. It assigns the responsibility of pruning the case-base to a case-base administrator who can use adaptation usage as one criterion in assessing the usefulness of a case in the case base.

6. CBD SYSTEMS FOR BUILDING DESIGN

Interestingly, the implicit use of CBR in building design has a long history. Experienced architects and engineers have always used their own accumulated experience while the inexperienced rely more on generative methods and external sources such as design reviews (Akin, 1986). Most architecture students learn to design using cases by analyzing, evaluating, and critiquing these.

The use of cost planning techniques in the quantity surveying profession (the British equivalent to cost engineers or construction cost controllers) provides another good example. In cost planning, cost analyses of previous construction projects are used to create a cost plan for a new project. The most appropriate analyses are selected, combined, and modified using heuristics, rules, and experience to create a cost plan. Usually, the cost plan forms the basis for future design development indicating budgetary allocations for each building element.

During the past few years many CBD systems have been developed for building design. Most systems deal with the structural design of buildings while a few handle architectural design. The following sections examine CBD systems developed for building design in chronological order. Space constraints have limited this review to 12 significant systems examined on the issues identified in Section 5.

6.1. CYCLOPS

This is considered to be the first CBD prototype developed in the Department of Civil Engineering at the Massachusetts Institute of Technology (Navinchandra, 1987, 1988, 1991). It was for landscape planning and design and provides assistance for the debugging of landscaping layouts. It combines constraint-based solution generation with case-based debugging and repair for the design of landscapes.

CYCLOPS uses ad-hoc models to represent cases as attribute-value pairs describing the problem situation as *unexplained-observation* and *explained-observations*, while solutions are *explained-solutions*. The case memory is organized as a flat structure allowing a serial search of the case base. Cases are retrieved in three ways: a *direct match* for an exact match, a *relaxed match* where matching is by similarity, and a *systematicity-based match* where matching is analogical and can span across domains. Analogical match-

ing is achieved by storing with a case, a causal explanation of the goals and subgoals of the case. CYCLOPS uses a problem-solving technique called *Demand Posting* to identify a problem, set up goals and subgoals, and to find appropriate cases and subcases needed to satisfy the goals and subgoals in an iterative process. Where direct solutions are not possible CYCLOPS uses the A* search algorithm to combine pieces of multiple cases using cross-domain analogy to provide innovative design solutions. New solutions created by the system act as precedents (new cases) for future solution generation.

CYCLOPS does not actively verify analogies. It is dependent on previous cases to find faults with an analogically modified case. It also assumes its causal relations are complete and that it carries the complete context of the situation. This assumption has broader implications on building designs as design elements are often interdependent and context sensitive.

6.2. STRUPLE

One of the earliest CBD systems for the structural design of buildings developed at Carnegie Mellon University (Maher & Zhao, 1987; Zhao & Maher, 1988). The system was originally developed as an expert system with an interface to a database containing previous designs. The aim being to reason from past experience.

STRUPLE's cases are represented as incomplete descriptions of building designs in numerous categories including: General Information, Geometric Information, Other Architectural Specifications, Load Information, Primary 3D Systems, Lateral 2D Systems, Floor Systems, and Foundation Systems. The design cases are stored in a relational database using the same structure. An intelligent interface is provided to the database to support retrieval of previous design solutions given a design problem described as several architectural and structural features of the design.

To retrieve cases STRUPLE uses a similarity metric based on a set of retrieval criteria classified as *required criteria, designed criteria*, and *no-match criteria*. These represent a relative prioritization of retrieval criteria. They also define allowable values for these criteria. The status of criteria is assigned by STRUPLE according to the description of the target building. A match score for a case is derived from the weighted summation of individual feature matches—zero reflects a perfect match and one reflects a perfect mismatch.

Design cases are presented to the user as attribute-value pairs. STRUPLE uses a design vocabulary that is applicable to a particular design and stored in its knowledge base as methods for transforming knowledge from existing cases to the current case. Each element is assigned a priority for evaluation. The subsequent synthesis process takes each subset of the design vocabulary, evaluates its suitability considering the priority levels and proposes alternatives to the user.

STRUPLE uses a fixed similarity metric to identify matches. This limits the capabilities of the system to a particular form of design solutions. Its analogical transformation of knowledge is considered only for structural systems and subsystems without considering the spatial layout or the number and location to these systems.

6.3. ARCHIE

ARCHIE is an intelligent case browsing system developed at the Georgia Institute of Technology (Goel et al., 1991; Pearce et al., 1992; Zimring et al., 1995). Its main aim is to provide architects with a design library for the conceptual architectural design of office buildings. The system was developed using ReMind. ARCHIE interactively supports the design task letting the user describe the problem and retrieving past designs providing suggestions and warnings.

Contents of a case in ARCHIE are categorized as *design goals* and *constraints*; *design plans* that specify building components and their configuration; *outcomes* that describe how well the plan satisfies goals; and constraints and *lessons learned* from each design solution.

Cases in ARCHIE are represented as attribute-value pairs in a flat record with more than 150 features. Features can be concepts, text, integers, real numbers, or functions. In addition to knowledge based on cases it contains domain models that capture the causal relationships between case concepts and primitive concepts such as objects, relations, and parameters of office buildings as a part of the language for representing and indexing cases.

ARCHIE uses the two retrieval mechanisms provided in ReMind, that is, nearest-neighbor matching and inductive knowledge-guided clustering. The first uses primitive concepts to retrieve building designs that satisfy a problem's goal and constraints. These concepts are hierarchically organized to specify the goals, plans, and outcomes of design cases. The knowledge-guided clustering approach uses simple domain relationship models to influence an induction algorithm that clusters cases into an index tree.

ARCHIE uses multimedia features to present cases to the user. Cases are associated to photographs, drawings, annotated plans, and animations. Users are allowed to move directly from design description to stories, from stories to problems, and between problems and responses. This is achieved through hypertext links.

ARCHIE does not use adaptation and it is left for the user to gain experience by using the knowledge they acquire to help them solve their design problems.

Many practical issues were raised by ARCHIE. Building design cases are complex, incomplete, and large in size. This requires a large vocabulary to encapsulate the design experience. Designers use multiple sources of knowledge, domain models, and rules in addition to experience. These needed to be reflected in CBD systems that aid designers. It also emphasized the need to have more practical user inter-

faces for CBD systems (especially for those intended for real use).

6.4. CADSYN

CADSYN is a CBD system for the structural design of buildings developed at the University of Sydney in Australia (Maher & Zhang, 1991, 1993; Maher et al., 1995). It is an implementation of a hybrid design process model using CBR with generalized decomposition and constraint satisfaction. It uses EDESYN an expert system that integrates rules and frames for problem decomposition and constraint satisfaction. The system was developed on a SUN workstation.

CADSYN's case memory is organized as case hierarchies and case indexing information. A case is represented in by a three-level hierarchy starting with the global context of the building, then describing the grid representation of each functional unit of the building, and finally descriptions of structural systems. A case in each layer is represented as a frame with attribute-value pairs describing features of the case.

Thus, the structural design of a building comprises a global context case (1st layer), several subcases based on function or geometry of the design (2nd layer). Each subcase in the 2nd layer has further subcases (3rd layer) describing structural subsystems that provide a solution to the 2nd layer case when combined together. Each subcase and supercase in the hierarchy is intereferenced for identification. Case contents were extracted from structural design drawings, documents, and design experts.

CADSYN uses a feature-based indexing scheme that uses subcase names to label the cases. A frame named *system list* (a set of slots representing systems in CADSYN) is used to store the names of all subcases. The generalized design knowledge base in CADSYN contains subsystems definitions, decomposition knowledge, structural design constraints, and procedural functions. These are implemented using a frame-based representation.

CADSYN's retrieval process retrieves a set of matching cases using the design problem specification. It performs a symbolic match to identify which subcases of a functional system should be considered for retrieval by examining the case lists in the *system lists* frame. Retrieved cases are ranked according to the number of features matched. The user is allowed to do the final selection of cases for adaptation.

A selected case is first adapted structurally through feature and value substitution. A constraint checking mechanism is used to evaluate this potential solution and to flag violations, which are then subjected to a modification process. A set of heuristic rules and the decomposition hierarchy are used for the modification. Where direct solutions cannot be found, problems are decomposed, solutions found for the subproblems, and subsequently recomposed using the constraint checking and modification process to provide complete solutions. New design solutions are added to the case-base and index modified by updating the *system list* frame accordingly.

CADSYN's indexing is flexible as indexing features are selected according to problem context. But the relative importance of features is set by the user. CADSYN highlighted the difficulty in formulating design cases from real designs. It also raised issues of expanding the domain-specific knowledge base for problem decomposition and incorporating domain-specific retrieval of cases. The constraint satisfaction approach assumes the problem specification is correct and appropriate. If not, this can lead to infinite loops in the iterative process of constraint satisfaction and modification. CADSYN gives little priority to design case presentation.

6.5. CADRE

CADRE is a CBD system for design adaptation and combination developed at the Swiss Federal Institute of Technology (Faltings, 1991; Hua et al., 1992; Hua & Faltings, 1993; Smith & Faltings, 1994). It supports the architectural and structural designs of buildings. CADRE stores geometric and symbolic information pertaining to architectural and structural design in a case. Cases contain more geometric information than structural information and are created from building plans stored as 3D CAD models. Presently CADRE contains a limited set of cases but ones that are diverse and rich in design concepts. It uses a fixed index of functional features to support case browsing and lets the user select a case for adaptation. Thus, the key issues this research deals with are design adaptation and combination. Another objective of CADRE is to integrate the different perspectives of the design team. They claim it has been achieved to a reasonable degree by integrating architectural and structural designs in one system.

A graphical user interface is provided and cases are presented to the user as CAD drawings on a main working window. The designer is allowed to change the free variables of the design and CADRE carrys out adaptation of other variables. Then it attempts to compute a solution by *dimensionality reduction*. If *topological adaptation* is required, it will generate possible topological alternatives, which are then shown in pop-up windows. A selected case is first inserted to the new site and to the new context. CADRE assigns weights to contextual features and the insertion is optimized by rotating and/or mirroring the case.

The dimensional adaptation process detects discrepancies between the new design and the site and converts these conflicts into parameters that are then resolved first. If this process is unsatisfactory, *topological adaptation* is triggered. This removes spaces in conflict and recreates those spaces ensuring harmony. Once this process is completed, the results are displayed and evaluated by the user. If unsatisfactory, the processes of *dimensionality reduction* and *topological adaptation* can be restarted.

Alternatively case combination is supported where spaces that conform to the present set of constraints can be found from other cases and combined. The adaptation processes above can then be reactivated to finalize the design.

CADRE contains two sources of knowledge: case-based knowledge (synthesis knowledge) in the form of previous design solutions and logic-based knowledge for analysis of context dependencies and to perform modifications. This type of knowledge is used in the *dimensionality reduction* and *topological adaptation* processes.

CADRE is implemented using Lisp and C with the interface provided by AutoCAD. Structural elements that are graphically presented are stored as object models. User interaction is through AutoCAD. Constraint posting is also carried out graphically within AutoCAD.

CADRE provides a method for adaptation that avoids the decomposition of the design. The hypothesis is that decomposition may disintegrate solutions losing implicit knowledge in a design while recomposition of a solution requires additional domain-specific and computational knowledge. It also raises the issue of the integration of the different perspectives of a design that they claim to achieve through dimensionality reduction and the geometric representation of cases. CADRE introduces case combination as a way to achieve innovation in CBD and to improve topological adaptation.

CADRE has a few limitations. It lacks a separate verification knowledge base and developers also recognize that its topological knowledge base is inadequate. The system is limited to adaptation of rectangular buildings due to the limitation of the representation scheme adopted. Work related to CADRE has been developed in the IDIOM system (Smith et al., 1995).

6.6. ARCHIE-II

This is the successor to ARCHIE and provides similar functionality but with improved usability (Domeshek & Kolodner, 1991, 1992, 1993; Domeshek et al., 1994). It is a story-based hypermedia browsing system that offers relevant documentation, stories, problems, and responses for perusal on a requested design topic in the domain of courthouse design. Stories, problems, and responses are organized graphically around design plans and provide multiple access of stories on the topic.

Indexing is based on artefacts, components, functional subsystems, and design issues and uses the nearest neighbor algorithm for case retrieval. The system does not provide for adaptation. The system was developed using DesignMUSE, a CBD shell originally developed for the implementation of a system for conceptual design of aircraft subsystems for Lockheed (Domeshek et al., 1994).

Stories represent detailed evaluations of specific buildings, while *problems* identify actual design goals and *responses* indicate the solutions to the problem (the synthesis knowledge). Design cases contain text, drawings, sketches, photographs, and other multimedia. Cases in ARCHIE-II are not a pure by-product of design as they have been augmented with evaluative material derived from explanations given by people involved in the design process. ARCHIE-II breaks a building design into chunks by looking for interesting outcomes of design issues in particular parts of the building.

Cases and subcases are interlinked and similar cases are linked through a fixed relationship index. A more flexible descriptive index allows searches for similar stories to the current story or problem. These support two types of searches:

1. artefact (building) retrieval based on user specified features, and
2. lessons (story or problem) retrieval based on user specified design interests.

Design stories (lessons) are indexed in five dimensions: design issues, building space, functional components, stakeholders (e.g., owner, builder, designer), and life-cycle phase (e.g., design, construction, use, maintenance). Cases are stored in memory as a flat library partitioned by the presentation type (e.g., story, design, description, problems, and solutions). It uses the nearest-neighbor algorithm to match and retrieve cases. Cases with the highest scores are presented to the user.

ARCHIE-II has improved on the interface and usability aspects of its predecessor. It has been completely converted to a multimedia case-browsing system that lets users focus on a particular subproblem of a design and broaden the search to similar cases to include related problems. It has addressed the issue of the complex nature of designs by segmenting a case into a number of subcases according to the lessons each case makes. It emphasises the need to provide alternative solutions to architects.

6.7. MEMORABILIA and PRECEDENTS

MEMORABILIA is a library of design precedents for architectural design developed at the Faculty of Architecture and Town Planning, at the Israel Institute of Technology [Oxman & Oxman, 1993*a*,*b*; Oxman, 1993, 1994]. The system specializes in the configuration of spatial organization in the schematic design of museums. It is primarily an intelligent case-browsing system for design. Hence, the research focuses on indexing and retrieval of cases. The first prototype of MEMORABILIA was implemented in HyperCard while future implementations are planned to be in common LISP interfaced to hypermedia.

Oxman (1993) defines a precedent as a case that denotes a unique idea. A precedent consists of situation descriptors (functional key concepts for the situation), solution descriptors (solution attributes, generation methods, and key con-

cepts), and outcome descriptors (solution evaluation and analysis). In addition, a precedent may be attached to a graphical representation. A frame-based formalism is used to represent precedent cases. Slots in the frame represent problem, solution, and outcome features.

An indexing tool provides two indexing methods that allow cross-contextual reminding. *Functional Key Concepts Indexing* uses design concepts such as orientation or circulation, to index cases within a building type. *Content Inference Indexing* also uses design concepts but only concepts that are in conflict. It finds solutions in which the posed conflict of concepts were solved. This indexing method is domain independent and cross-contextual. The advantage of this method is cases may be searched on how a particular kind of conflict was solved; this is a more directed search as opposed to an exploration of solutions.

The system retrieves appropriate cases as solutions to the problem description of a current design. Problem description includes building type, site type, site conditions and other contextual descriptors (e.g., required architectural style). Cases retrieved can be viewed as stories or as graphical representations.

Ongoing research has further expanded the concepts described above in a system called PRECEDENTS (Oxman & Oxman, 1993*b*; Oxman, 1994, 1995). It decomposes a precedent case into design stories. A design story is an annotation of the conceptual design that characterizes the uniqueness of a specific design precedent (i.e., a chunk of knowledge emanating from a precedent). A tripartite representational schema is used to describe a design story (i.e., design *issues* of the problem, a particular solution's *concept* and related *form* descriptions). These generally map to the problem, solution, and outcomes in MEMORABILIA. PRECEDENTS' case memory uses a semantic net based on the domain content vocabulary. From each node of the net stories can be retrieved and hence the precedent case. Issues, concepts and forms, as well as the precedents themselves, can be interlinked.

PRECEDENTS uses two types of index. Search indexes for directly finding relevant solution principles and design solution (stories) and browsing indexes for broader searching using hard coded linkages between design concepts within the semantic network. Within each type cross-contextual searching and browsing is allowed.

MEMORABILIA and PRECEDENT consider architectural design in terms of formulation or configuration of functional spaces which is appropriate to the conceptual design stage. As an intelligent case browsing system, they can refresh or enlighten the designer with remindings from the past. The way to reuse the knowledge gained from the reminding lies in the hands of the user.

6.8. FABEL

FABEL is a major research project for the design of complex buildings with a high degree of technical installations (FABEL, 1993*a,b,* 1994; Bakhtari & Bartsch-Sporl, 1994; Bartsch-Sporl, 1995). FABEL is the AI-based support system for the A4 model prototype design system based on the MIDI & ARMILLA design methodologies for complex industrial buildings (Haller, 1974, 1988). The project has the broad objective of the seamless integration of case-based and model-based reasoning in design and was first implemented as a CBD support system. The system deals with the detailed architectural design of buildings, decomposing the design into design elements or segments where CBD support is provided to the user when required within a CAD environment.

FABEL's cases have many representations. Cases form instances of object-oriented models (symbolic representation) and are also represented as CAD images. This allows the user to switch between representations when required. The symbolic representation works with prototypical cases. Whenever a CAD image is required and prototype parameters values are given, a CAD construction program transfers these objects to CAD representations enabling either 2D or 3D visualizations.

The project has identified the development of retrieval methods that use semantic information as well as structural information as one of their major challenges. To this extent they have devised five methods for identifying the similarity of technical drawings. Three of these have been implemented as separate modules and can be used in combination if required (FABEL, 1994, 1996).

1. An associative memory (ASM): The cases are indexed according to type and attributes of technical objects and according to their frequency of occurrence. An associative memory is used for similarity comparison (Grather, 1994).
2. Distance measures (RABBIT): The cases are indexed as above but similarity is identified by distance measures (Linowski, 1994).
3. Raster displays (ODE): This method compares raster displays on different levels of grain size (i.e., abstracted bitmaps of design pictures). It is a method primarily based on the graphical appearance of cases (Coulon & Steffens, 1994).
4. TOPO: uses a topological representation of objects in a design. It uses the topological relationships to compute an edge-graph representation of the layout and searches for the largest common subgraph (Coulon, 1995).
5. Gestalts: uses recurring patterns (e.g., herring-bones) to classify layouts. Ten different categories have been identified (Schaaff, 1994; Schaaff & Voss, 1995).

A retrieval strategy that uses these retrieval algorithms has been created called ASpecT (Schaaff & Voss, 1995). Of these methods, the first three have been tested and others are in various stages of development and testing. The FABEL pro-

totype uses a two staged retrieval methodology (Genter & Forbus, 1991), which makes an initial retrieval of candidate designs and subsequently selection is narrowed with the use of computationally more expensive methods, such as numbers 3 and 5 above.

FABEL supports adaptation in many ways with the incorporation of specific adaptation tools. These task-specific and domain-specific tools use their own knowledge-based systems to support adaptation. Most of these adaptation tools are in the stages of development and testing. A degree of adaptation has been achieved through TOPO, which topologically transforms a selected case to the present situation. Other tools are used for specific adaptation purposes such as modifying the arrangement of columns in a steel framed structure (AAAO) and adaptation of return air-duct connections (SYN*). These adaptation tools use domain specific knowledge in the form of rules used to satisfy constraints generated from the case transformation process. Some of these tools are incorporated with verification knowledge while other separate verification knowledge-based tools are being formulated (e.g., ANOPLA for the verification of pipes on a grid layout and CHECK for the verification of topological relations of adapted objects).

FABEL's main contribution is in the formulation of different retrieval strategies that allow cases to be retrieved for a particular view or design perspective. In addition, it provides a method to directly and graphically match CAD layout drawings (stored as cases) for assessment of similarity and subsequent retrieval. Additional information and a full list of publications related to FABEL can be found at the FABEL web site (FABEL, 1996).

6.9. SEED

SEED supports the early design of buildings (Flemming, 1994; Flemming et al., 1993, 1994). It integrates many design generative and representation systems with CBD. It is implemented for the architectural design of buildings and was developed at Carnegie-Mellon University. It is a major project funded by several U.S., Australian, and Danish organizations. SEED contains 3 modules:

- Module 1: Architectural programming for conceptual design development and briefing.
- Module 2: Schematic layout design.
- Module 3: Schematic configuration design.

It uses generic systems for design generation and representation and each module is supported with a CBD component, thus allowing the rapid development of design alternative. Its aim is to use system-generated designs as cases for reuse. Its case memory accumulates as a side-effect of a design firm's normal design activities.

SEED's modules support the creation of the problem specification, generation of solutions, and evaluation of the solution, which forms the key contents of a case. Cases are instances in a structured object hierarchy primarily based on a part-of relationships. These relations are transformed into spatial containment relations in a solution. Constraints on the relations are expressed as separate objects, which trigger tests to check the transformation.

SEED's problem specification is divided into two central constructs:

- *Design units* are the basic spatial and physical entities that describe shape, location, and nongeometric attributes.
- *Functional units* prescribe the design units needed to satisfy a given context. These may contain a hierarchy of constituent functional units.

A problem specification is a structured object described in terms of attribute-value pairs. Case retrieval is carried out mainly on the structural similarity of cases. Matching is performed on classes and if required on subclasses. The total match score for a case is achieved through a weighted sum of matching attributes. The designer is allowed to set the weights.

SEED allows the development of the problem specification through an index refinement process. They consider that the design problem evolves with its solution. Cases retrieved are ranked accordingly and presented to the user for evaluation and selection. Index refinement is then used to adopt a selected case (i.e., to refine the problem specification). The evaluation objects would then test the partially adopted problem specification. The evaluation can be carried out before, during, and after adaptation. Adaptation is carried out on a graphic (CAD-like) window and primarily deals with functional space allocation. Adaptation is supported by a range of automated commands (e.g., add, remove, edit, and generate). Interactive adaptation by the user is also supported.

SEED is under development and they are examining numerous methods for case indexing and retrieval. They are also investigating ways of incorporating or attaching design notes, hints, and the like to cases. They plan to store cases in an object-oriented database with a separate case-base representation that would provide a platform for experimenting on different indexing methods. They claim "*bringing research design methods and technology closer to practice*" as one of the primary goals of the research.

6.10. CASECAD

CASECAD is a design aiding CBD system for structural design of medium rise buildings at the University of Sydney and the University of Wollongong, Australia (Maher et al., 1995; Maher & Balachandran, 1994*a,b*). It is implemented on SUN SPARC workstations. It uses the X view tool kit as its interface builder, along with C and common

LISP plus FRAMEKIT. In addition, AutoCAD and XFIG are used to display graphic files.

CASECAD contains several modules: case memory (CMM), containing model-based and case-based knowledge; a case-base manager (CBM), supporting the creation, browsing, modifying, displaying, and saving of cases; a case-base designer (CBDr) that retrieves and ranks cases; plus CAD packages that support the graphical presentation and editing of cases and a graphical user interface to interact with all the modules.

Case memory organization in CASECAD primarily consists of design models that encapsulate generalized domain knowledge and provide an organizational schema for the case memory. This object-oriented classification hierarchy contains different levels of abstraction. The structural design problem is divided into subproblems (e.g., grid systems, training systems, core-structure, floor systems, etc.) in a functional decomposition structure. Flexibility in indexing and retrieval is achieved through the categorization of attributes at each level of abstraction (i.e., a class) into function, behavior, structure conforming to the schema proposed in the design prototypes (Gero, 1990). In addition, relation attributes define the relationship of class to its superclass. Cases are represented as instances of classes encapsulating specific design knowledge.

A case in CASECAD is a single building and a subcase is a structural component of the building. Both cases and subcases are instances of their respective classes and subclasses in the design model. A case contains attributes that describe the case along with attributes that attach CAD files (2D and 3D) to the case. CASECAD has the ability to index cases separately on function, behavior, and structure or all these together.

CASECAD incorporates an iterative process for case retrieval. Retrieval is carried out in two stages. First, a set of attributes is retrieved, and second, attribute-value matching is carried out. Thus, cases are ranked according to the number of attributes that match. The similarity of attribute values is not considered.

CASECAD has created design models to represent domain knowledge and cases to imply specific instances. It has successfully integrated CAD images with cases and its indexing and iterative problem specification development provide very useful features.

6.11. ADA

ADA is an interactive design system for architectural design developed at the University of Wales and the University of the West of England (Hunt & Miles, 1995). It provides a repository of architectural designs with annotations that help to elucidate the intent of the designer along with methods for evaluation of the design. The ADA initial prototype was developed in ReMind and a second prototype was implemented in POP-11 within the POPLOG environment.

ADA implements a design as a hierarchy of subcases based on functional spaces in a design. Thus, design can be considered as a complete solution or a partial solution (i.e., solution to a design subproblem). The hierarchy is an object-oriented representation of the design. Cases are stored as plain ASCII files and are read into ADA, which constructs the object-oriented representation of the case.

CASES in ADA contain not only geometric and functional information on the design, but also the rationale for the design, including its history, justifications, goals influencing the design, and other annotations. These are implemented as links from the object representation. These multimedia features are provided to give a broader understanding of the design.

ADA performs two types of matches. The first is direct matches that consider all attributes associated with a design and their links. This search records all terms that match the initial request. The second type of match is a nearest neighbor match, which uses the hierarchy to classify cases. The weighting system gives these matches half the weight of those that have been directly matched. An algorithm ranks cases according to the highest match score.

Retrieved cases can be viewed as text documents, CAD layouts, and additional annotations. The user is allowed to select a case for adaptation. However, ADA does not provide automatic adaptation. They argue the implications of modifications to a design are complex, numerous and moreover not preferred by the designers themselves. A modification may have affects or consequences on aesthetic, artistic issues as well as on cost or even corporate image. They believe that managing these interrelations is an impossible or prohibitively expensive task.

However, ADA does provide a design repair and evaluation module. Evaluation is carried out on the modified design using heuristic rules extracted from a design manual and by the use of existing algorithms such as space allocation algorithms. The evaluation knowledge base can check the user-modified design for any violation of rules or parameters. They also envisage the use of design concepts to assist in the modification and evaluation process. The repair module fires a rule if its parameters are violated and prompts the user with suggestions on how to alter it.

ADA supports an iterative design development process. Cases can be retrieved, modified, or combined at any time during design development. The evaluation module can be evoked to check the validity of adaptation. Thus the process is user centered and user assisted.

A potential problem in retrieval of cases in ADA is its halving of match weight for all partial matches. If the partial match is on an important feature a case may still get a lower score than one where less important features have an exact match. The prioritization of features on match alone may result in some structurally less similar cases gaining a higher match score.

Conceptually, ADA is a usable system for architects that frees them from mundane tasks such as finding a suitable

previous solution and assists them in creative and knowledge intensive tasks such as constraint checking.

6.12. NIRMANI

NIRMANI is an interactive CBD system developed by AI-CBR at the University of Salford (Perera et al., 1995; Watson & Perera, 1995; Perera & Watson, 1995, 1996). The system is for the conceptual architectural design and estimating of warehouse buildings. The prototype is implemented in ART*Enterprise, which provides an object-oriented knowledge-based development environment with methods, rules, and case-based reasoning. It also uses AutoCAD, Netscape, and other applications for the presentation of associated multimedia.

NIRMANI integrates estimating and design within a case-base environment. It provides a case memory organizational structure and four dynamic case indexing methods for case retrieval. Adaptation is supported by a modification knowledge base and verification rule base. It also contains a module for case-base maintenance.

A building design in NIRMANI is a meta-case consisting of a hierarchy of cases and subcases. At the top of the hierarchy is the *Project Context* case. The second level contains *Architectural Context* and *Estimating Context* cases representing the perspectives (or views) of architects and cost estimators. A third level decomposes the architectural design into functional spaces and aesthetic requirements hierarchies and the estimating problem into an elemental classification hierarchy. Cases are stored as records in a relational database external to the system. A unique case name is used as the key for identifying subcases of the same design. An object hierarchy within the system maps to records in the database and cases are presented (when required) as instances. Cases contain attribute-value pairs as case features describing the respective design problems.

Each case feature can have multimedia documents attached to it. These are text documents (cost reports, design annotations, user evaluations, etc.), photographs, 2D and 3D CAD, video clips, pointers to spreadsheets, and even WWW pages. All multimedia feature details are stored in a separate database.

A *Project Context* case describes the environment within which the project was carried out (features such as the site condition, details, type of building, and other features common to both perspectives). The second level cases (architectural and estimating) describe the context of the subproblems. The system prefers subcases to be retrieved with the same context to reduce problems of case adaptation due to contextual dissimilarity.

NIRMANI provides four indexing methods using nearest-neighbor matching for case retrieval. Its *default index* contains all cases in the case base. This default method will not necessarily retrieve cases with similar project contexts. The other indexing methods of *narrowing dynamic indexing, partial dynamic indexing*, and *hierarchical dynamic indexing* restrict case retrieval to similar project contexts and architectural or estimating contexts, respectively. A case-base weighting system and case-attribute weighting system is used to derive a match score. The weighted and normalized scores of cases are used to rank cases for each case base and to rank case retrieval from all case bases.

The interface of NIRMANI allows cases to be viewed as attribute-value pairs, CAD drawings and other multimedia elements. It supports parallel case comparison using a tabulated form. Users are allowed to select cases for adaptation.

Adaptation in NIRMANI is carried out in two phases. Primary adaptation adapts and refines the problem specification. An index elaboration and index revision mechanism is used for this purpose. Secondary adaptation converts the problem specification into a solution through a case modification and combination process. A modification knowledge base, containing a set of functions, supports case modification. A heuristic rule base is used to verify modifications and combination effects in the solution.

New solutions developed contain a list of cases that contributed to them letting a case-base administrator assess the usage of cases and thereby decide if pruning of the case base is required. Case-base maintenance mode allows editing, addition of new cases, and removal of cases from the case base.

One of the main features of NIRMANI is its ability to acquire cases from existing databases. This has been successfully demonstrated using estimating data. However, the absence of similar data for architectural design required design data to be extracted manually from drawings and documents. The present implementation provides secondary adaptation only for the estimating perspective. However, future research on NIRMANI would involve it being integrated with a knowledge-based design system called SPACE (Alshawi, 1995) that uses AutoCAD and an underlying object model to represent design knowledge. This would let NIRMANI achieve detailed architectural design adaptation.

NIRMANI identifies knowledge elicitation as one of the key phases in the development of a CBD system. This is required not only to formulate adaptation knowledge, but also to create the case memory structure. Identifying what features are important to be presented in a case and their relative importance is a difficult task. We found that the determination of plausible weights for case features is a black art. This is because these weights are highly context and perspective dependent. Hence, in NIRMANI setting feature weights can be left to the user. Another related issue arising from this is the number of case attributes to use for case retrieval and the number of case features (attributes) to present to the user. Again, NIRMANI lets the user select which features should be included in the index for retrieval but it presents all features of a case to the user. However, there are arguments against this approach because it can overload the designer with detail (Lehane & Moore, 1996).

7. COMPARISON OF THE REVIEWED SYSTEMS

CYCLOPS and STRUPLE pioneered the use of CBR as a technique in design problem solving. CYCLOPS evolved as a tool for innovative design, and explained the use of experience across domains to generate new design solutions. It showed the way to solve problems with cross domain analogies where direct previous solutions cannot be found. STRUPLE opened up another avenue in CBD by integrating CBR with an expert system. It showed how domain-specific knowledge can be used to evaluate and transform knowledge gained from previous design solutions to similar design problems.

These systems raised many issues that need to be addressed by CBD systems. CYCLOPS identified the need to go beyond case-based verification of solutions and the need to identify interactions between design components. STRUPLE points out to the need to have more flexible strategies for retrieval and the need to make use of spatial configuration of previous design solutions in deriving new design solutions.

ARCHIE aimed to aid the designer by reminding how design problems were solved in similar situations. It used multimedia features to enhance the understanding of design problems and their solutions. It confirmed issues raised by CYCLOPS and STRUPLE by highlighting the need to supplement case-based knowledge with other types of design knowledge. It showed that designs should be decomposed into manageable chunks and that not surprisingly users need usable interfaces.

CADSYN is a hybrid system that combines rule-based, model-based, and case-based reasoning. It finds design solutions for a given problem specification using decomposition knowledge and constraint satisfaction with minimal user interaction. It provides explanations of its reasoning steps as does a classic rule-based system. As with ARCHIE, it also emphasised the complexity of real design cases. It indicated that drawings alone are inadequate for developing a case base and that these need to be augmented and commented by the designers. It also identified the need to expand the domain knowledge base to provide different retrieval and adaptation mechanisms.

CADRE, contrary to most CBD systems, emphasizes case adaptation and lets the user retrieve and select cases. It takes the opposite approach to CADSYN in terms of design adaptation. It avoids decomposition and recomposition of design solutions. Instead, it uses a domain-independent dimensionality reduction process and domain-dependent topological adaptation process. In line with many other systems, it provides a CAD interface for design adaptation in which a graphical representation of the design is supported by an underlying object structure. They argue against expensive adaptation which would destroy implicit features of a design. CADRE also considers context sensitivity of design cases when using them in new situations. It also raised the importance of the integration of different perspectives of the design (i.e., architectural, structural, services, etc.).

ARCHIE-II addressed some of the issues raised by its predecessor. It addressed the issue of complexity and information richness in building designs by segmenting cases into chunks or subcases according to the lessons they teach. By incorporating two indexing methods and two retrieval strategies, they have achieved a greater degree of flexibility in the retrieval of cases. Hard coded relationship indexes capture the domain-specific knowledge that defines the relationship of one case to another. With the help of constant consultations with prospective users, ARCHIE-II has achieved greater usability than its predecessor. It raised new issues of collecting cases for incorporation into the case base. They have devised some methods for data collection (style sheets) and are experimenting with these methods. However, it is not clear in what way they are going to handle case maintenance issues in the expanding case base.

MEMORABILIA is similar to the ARCHIE systems in that it aims to provide an intelligent case-browsing system for the conceptual design of buildings. However, its main aim is to provide cross-contextual reminding. Conceptually, this is similar to CYCLOPS, which achieves it through analogical reasoning across domains. But MEMORABILIA attempts to achieve it across different design contexts by looking for similar solutions in different building types. Alternative indexing methods let cases be retrieved on conflicts in design concepts. This allows users to identify how competing concepts in designs were solved in previous situations. This is an important issue that other systems deal with mainly in terms of design constraints. PRECEDENT, the successor to MEMORABILIA, decomposes design precedent cases into chunks of knowledge represented as stories. Thus, PRECEDENT's main difference to MEMORABILIA is in the way in which the case memory is organized. This is similar to ARCHIE-II's design story organization. However, PRECEDENT classifies a story as design issues, concepts, and forms organized as a semantic net with links to similar issues. Its other main feature lies in the addition of goal-directed search and broad issue-, concept-, or form-browsing facilities along with cross-contextual reminding in search and browse modes. MEMORABILIA's main contribution lies in indexes for cross-contextual reminding while PRECEDENT is the organization of cases as stories connected to design precedents.

FABEL creates cases in a similar way to CADRE, that is, its CAD layout drawings have an underlying symbolic representation in terms of objects. This has enabled them to create cases as instances of prototypical object structures. FABEL's major contribution lies in the development of multiple case retrieval strategies, facilitating multiple views of the same data space. In our view, the most import and unique of these is the strategy to retrieve CAD layouts based on their structural similarity. Retrieval strategy using gestalts,

layout fragments containing recurring patterns, is another unique approach. The method that represents topological relations to graphs is a somewhat similar approach to CYCLOPS's influence graphs. The case retrieval shell, which encompasses the different retrieval strategies of FABEL, not only provides a common case base but supports different views of the same cases.

SEED provides case-based design support throughout the design life-cycle, although schematic design remains their immediate concern. SEED's case representation is similar to that of CADRE and FABEL; that is, a graphical representation with an underlying object structure. At the same time, they endorse ARCHIE's (I & II) ability to annotate design. SEED raises a new issue in building design; namely, the need to develop the problem specification and its solution in an iterative process (i.e., index refinement). This is important in building design, as the development of the design brief is an evolving process. SEED identifies the need to identify relative weights for case features. However, they point out that predetermination of such weights may lead to the problem of mutual preferential dependence of features (Keeney & Raiffa, 1976). Therefore, they let the user set weights for features according to the needs of the retrieval. Like FABEL, SEED is limited to the adaptation of rectangular spaces and stresses the need for adaptation to be interactive.

CASECAD's main contribution is the integration of CAD with CBR for case presentation. This allows the designer to view a selected case or subcase in a familiar way. In this respect, it is similar to ARCHIE-II and PRECEDENTS, but CASECAD goes a step further by allowing the user to use a case in the CAD environment to directly evolve a new solution. CASECAD does not provide an adaptation knowledge base. However, it does augment the need for case combination by functionally decomposing the structural design. Subsequent adaptation by combination or otherwise is left to the user. A danger in this approach could be the combination of subcases out of context.

CASECAD provides a means of indexing and retrieving CAD drawings, but at the expense of an introduction of additional features to a case. CASECAD's use of design models and its integration with the case memory provides a useful paradigm for case memory organization. This can be compared with FABEL and SEED's proposed approach. These use a domain model to hierarchically classify design elements into classes where instances of classes denote cases. Similar to SEED, it identifies the need to develop the problem specification in an iterative process and goes one step beyond with the incorporation of index elaboration in addition to index revision. Similar to ARCHIE, FABEL, and MEMORABILIA, CASECAD also identifies the difficulties related with case acquisition—namely, that drawings and design documents alone do not provide adequate information for the formulation of a design case.

ADA is an interactive CBD system with a case content similar to ARCHIE-II. But, ADA rallies all subcases around a design case as opposed to design stories in the case of ARCHIE-II or PRECEDENTS. It is similar to CASECAD in terms of organization in that designs consists of a network of subdesign solutions and are implemented as an object hierarchy. Subcases can be assessed independently or collectively. Like ARCHIE-II it provides annotations to subproblems of design. ADA provides a verification knowledge base to check user modifications and guides the user in the direction of a plausible solution. In terms of design verification knowledge, ADA uses heuristic rules and space allocation algorithms similar to SEED and CADSYN. But the verification knowledge is used to verify user adaptation and not system adaptation as in the case of CADSYN and CADRE. ADA's uniqueness is with its conceptual approach to interactive design, assisting the designer in mundane but highly computational tasks. They attempt to achieve a balance between usability and the degree of automation. In terms of case content they highlight the need to provide information as to the development of the design in terms of design annotations conforming to the views expressed in AskJef (Barber et al., 1992).

NIRMANI primarily addresses two issues: the need to provide different perspectives or views of the design and the need to retrieve cases sharing a similar context to ease case adaptation. It uses a case hierarchy to integrate two design views: those of the architect and the cost planner. FABEL uses the ASpecT multiview retrieval strategy to achieve the same end. However, NIRMANI's approach does create some data redundancy in the case bases. In a similar way to NIRMANI, FABEL's ASpecT retrieval strategy has the user set weights for features (aspects) to define context. However, FABEL uses a neural network assisted mechanism to set weights (Schaaff & Voss, 1995) while in NIRMANI the user sets the weights.

NIRMANI is similar to CASECAD in terms of case memory organization; both use domain models as class hierarchies and instances of classes representing specific cases. NIRMANI allows designers and estimators to reason with confidence using the same reasoning paradigm. NIRMANI uses multimedia to present cases to the user in a similar manner to CASECAD. However, they avoid burdening the case content by providing pointers to media documents and it uses SQL-like queries to retrieve media documents. Using a similar approach to SEED and CASECAD, NIRMANI develops the problem specification using index elaboration and revision techniques. For the formulation of adaptation knowledge, it takes a similar approach to CADSYN, using a modification knowledge base and a verification rule base. NIRMANI raised the issue of providing multiple perspectives of a design case along with the importance of design context in case retrieval and adaptation. It identifies the difficulties of identifying case features for design problems and the problems of setting weights for the features. It claims that the retrieval of cases from a similar context eases adaptation by reducing constraint violations.

Table 3. *Summary of the comparison of CBD systems*

Name	Domain	Brief Description	Representation
CYCLOPS	Landscape design	First case-based system in design. Cases stored as problems and solutions for new problems achieved within a given set of constraints. It is used to debug and repair landscape layout designs. It uses cross domain analogies to provide new solutions to its problems.	Uses ad-hoc models to represent cases as problems (explained and unexplained observations) and their explained solutions in a flat case memory structure. These provide indexing features.
STRUPLE	Structural system design of multistorey buildings	Developed as a rule-based estimating system with an intelligent database interface for reasoning from previous experience. Previous design cases are stored in a relational database. Retrieval criteria are weighted to indicate relative importance of features based on domain-specific knowledge. A synthesis process is used to evaluate all subsets of a chosen design.	Design cases are represented as attribute-value pairs mapped to a relational database. It uses a flat organization of the case memory.
ARCHIE	Architectural design of office buildings	An interactive CBD aiding system implemented using ReMind. Users specify their problem description and the system retrieves and displays past designs and provides suggestions and warnings. In support of evaluation, the system computes potential outcomes and retrieves and displays past designs with similar outcomes. Its indexing vocabulary is based on design goals, outcomes, and situation descriptors. It uses nearest neighbor and an inductive retrieval algorithm.	Cases are represented as flat frames describing case features as attribute-value pairs. Design cases contain design goals (problems) outcomes (description of what were the solutions) plans (drawings, sketches, graphics etc.), and lessons (how solutions were achieved).
CADSYN	Preliminary structural design of multistorey buildings.	A hybrid CBD system developed on SUN stations in common Lisp and FRAMEKIT. Hierarchical case memory with attribute-value pairs. Uses a decomposition knowledge base and design constraint representation for retrieval of subcases. Case combination uses a constraint satisfaction approach for solution recomposition.	A design solution is decomposed into a set of hierarchically organized global cases and subcases. The hierarchy consists of three layers: global context, grid representation, and a hierarchy of subcases describing the design solution for each grid system.
CADRE	Architectural and structural design of buildings.	A CBD system emphasizing case adaptation. Cases stored as design drawings. User selects an appropriate design for adaptation. CADRE contains knowledge in two forms. Case-based knowledge and logic-based knowledge that is used for adaptation of cases, contextually, dimensionally, and topologically. Implemented in Lisp and C to represent CAD models of buildings as objects. AutoCAD provides the GUI.	Cases are design plans, containing more geometric information that structural information (i.e., cases stored as 3D CAD models).
ARCHIE-II	Architectural design of court buildings.	An extension of ARCHIE. Provides a multimedia case base for browsing of similar design problems and their solutions held as design stories. It uses a flexible descriptive index for user directed searches and a relationship index to link similar cases. Cases can be retrieved interactively and sequentially leading from one problem to another or diversifying to different issues.	Building design contributes to many design subcase segments that are created around a design lesson. Uses a flat case memory partitioned as: stories, problems, and solutions.
MEMORABILIA & PRECEDENTS	Space layout design of buildings.	A prototype system developed using HyperCard stores design stories that have the status of precedents. The user is presented with an appropriate design story and a corresponding design precedent that matches the design problem. The system in addition, allows cross contextual browsing. Presently contains museum design precedents.	Design cases represent design stories consisting of design issues, concept, and form cases. Represented as frames with slots (representing features that describe design issues, concept and form). Fillers provide values identifying each story.
FABEL	Architectural design of buildings.	A major CBD project for complex industrialized buildings with a high degree of technical installations. Uses the ARMILLA & MIDI design methodology and the DANCER design tool to generate cases for the system. Aims to integrate CBR and model-based reasoning in the design tool.	Cases are primarily represented as CAD layouts with an underlying object structure.
SEED	Schematic architectural design of buildings.	A design generation system consisting of three modules: Module 1: Architectural Programming Module 2: Schematic Layout Design Module 3: Schematic Configuration Design. The concept of *design units* and *functional units* are used for defining the problem and the case features.	Cases are instances of a class hierarchy. Cases contain problem specification, solution (design), and outcomes (evaluation).
CASECAD	Conceptual structural design of medium rise buildings.	A case retrieval system incorporating a multimedia approach. Cases are represented in an object hierarchy and as 2D and 3D CAD images.	Design cases and subcases form instances of an object hierarchy based on a functional element decomposition of structural design (design models). Case hierarchies contain attributes to describe associated media files (CAD drawings).
ADA	Architectural design or buildings.	Interactive CBD systems for storage, retrieval, and evaluation of architectural designs. Cases contain drawings and annotations expressing intent of the architect. Implemented in POP II.	Design of a building is contained in a meta-case as a collection of subcases. Cases are represented as an object hierarchy based on a decomposition of design on functional spaces.
NIRMANI	Conceptual architectural design and estimation of light industrial buildings.	An interactive CBD and estimating system uses a multimedia case base stored as a relational database mapped to an object hierarchy in ART *Enterprise. Provides case-base storage, dynamic indexing, retrieval, and adaptation of cases. Adaptation is user centered but guided by adaptation knowledge.	Uses a multiple-perspective hierarchical case-base structure with 3 layers: project context, design perspective, and problem decomposition. Cases are instances with attribute-value pairs to describe features.

Retrieval	Presentation	Adaptation	Other/Comments
Cases are retrieved as direct matches, relaxed matches (abstract match), and systematically-based match (using cross-domain analogies). Uses a serial search to retrieve cases that match indexed features.	Cases are presented as precedents in solving a previous problem.	Provides case-based solutions for new problems. It can combine cases to generate innovative solutions.	New solutions are verified using the cases in the case base. Emphasis on retrieval and adaptation.
Retrieval is based on a fixed similarity metric that compares significant common aspects of the matched case to the current case. Matching is on weighted features.	A list of ranked cases is presented to the user.	Uses the design vocabulary of selected previous designs to generate a new solution. Instead of directly taking the previous solution it compares the selected cases design vocabulary for each element with its knowledge base and identifies the most promising for use during the synthesis process.	It combines the use of case-based knowledge with domain-specific knowledge in an estimating system. The fixed similarity metric limits the capabilities of the system. It only considers the structural system and not the spatial solution. Emphasis is on retrieval and adaptation.
Two retrieval strategies: 1. Nearest neighbor based on primitive concepts. 2. Qualitative retrieval based on inductive knowledge-based clustering using simple domain models.	Cases are associated with text, drawings, annotated sketches, images, animation, etc. Case browsing can be done as index-based browsing and hypertext-based browsing.	None.	Highlights the complexity of building design cases. Need for user friendly GUI. Need to incorporate domain-specific knowledge for reasoning.
A feature-based index of cases (and subcases) is created as a frame (the system list). Cases and subcases of each structural subsystem are recorded in the index. Cases are retrieved and ranked according to the number of features matching.	Envisage minimal user interaction. However, provides a trace of the reasoning process in the form of how cases were adapted.	Cases and subcases are first structurally adapted (feature based substitution), constraint violations are checked, and a modification process assisted by heuristic rules and decomposition knowledge then modifies violated constraints.	Highlights the complexity of design cases. Provides flexible indexing. Lack of domain knowledge for ill-conceived problem specifications may lead to infinite loops or no solutions. Needs to expand domain specification knowledge for constraint satisfaction as well as case retrieval.
User selects an appropriate case.	Cases are presented as design plans. Users are allowed to modify the design in order to satisfy constraints.	Adaptation uses a dimensionality reduction process to overcome conflicts. Adaptation is carried out to satisfy structural and space layout problems. Cases are combined where appropriate to generate new solutions.	Takes a case adaptation approach that avoids decomposition and recomposition of a design solution. Identifies the need for integration of design perspectives. Dimensionality reduction needs to be carried out for each adaptation and combination. Limited to rectangular spaces. Requires more maturity in topological adaptation knowledge.
Cases can be retrieved as a whole design (a building) or a design story dealing with a subproblem of a design. Users are allowed to browse through the case base by focusing on a problem or by broadening the search. Uses nearest neighbor for matching and retrieval.	Cases are presented to the users as text, drawings, annotated sketches, and photographs.	None.	Emphasis is on the development of a design aiding system. Thus, priority is on case presentation and the definition of reusable chunks of a design.
Provide goal directed searching and broad-based browsing. Cross-contextual reminding is used within search and browse modes.	Cases or stories are presented as story cards in a Hypermedia System. Precedent cases are graphically illustrated.	None.	Main contributions of MEMORABILIA is the indexing methods that allow cross-contextual reminding. PRECEDENT's contribution is in the use of a semantic net of stories connected to precedents.
Provides five main case-retrieval strategies based on: Associative memory (key words), attribute vectors (attribute value pairs), topological similarity (graph based), gestalts (similar patterns), and object density maps (graphical similarity using bitmaps). ASpecT encompasses all these retrieval methods in one environment.	Presented mainly as CAD layout drawings and bitmaps.	Topological adaptation allows the transformation of topological features from a selected case. Other methods being developed use adaptation knowledge stored as rules and a constraint satisfaction approach.	Main contribution lies in providing a multitude of case-retrieval methods. Recognizes the need to allow multiple perspectives of the same data.
Cases are retrieved by traversal of the class hierarchy. Problem specification is developed through an index refinement process.	CAD/graphical user interface used for case presentation and adaptation.	An interactive user centered adaptation approach is envisaged. Uses many adaptation primitives (add, edit, copy, etc.). A constraint-checking mechanism provided.	Cases are a byproduct of design generation. Adaptation is limited to rectangular spaces. Raises issues of problem specification development.
Indexed in two levels. 1. Category Indexes that have Relations. Functions, Behavior, and Structures as four main categories. 2. Attribute Indexes—many attributes per category. Flexible indexing and case retrieval. Provides index elaboration and index-revision techniques for problem specification development.	Cases are presented as attribute-value pairs and/or as 2D & 3D CAD images.	Allows manual adaptation of cases and the addition of new cases to memory.	Enables flexible retrieval of cases where adaptation is left to the user. Useful integration of CAD with CBR and design models with CBR.
Cases can be retrieved individually or collectively. Retrieval based on exact match and partial match.	Cases can be viewed as drawings, design concepts, histories, justifications, and annotations to design cases.	The user adapts a selected case and a verification knowledge base checks the validity of the modifications and prompts the user for appropriate action.	Tries to achieve a balance between the degree of automation and usability.
Cases are retrieved interactively using one of four nearest-neighbor based indexing methods. These retrieve cases with identical or similar contexts. Index elaboration and revision are used to develop the problem specification.	Cases are presented as attribute-value pairs, individually or in a table. 2D and 3D CAD drawings, photographs, video, documents, and WWW pages can be associated with case features.	Adaptation knowledge is stored in a modification knowledge base as functions and as rules in verification knowledgebase. Adaptation is carried out in 2 stages. Primary adaptation refines the problem specification using index elaboration and revision. Secondary adaptation modifies case contents and combines cases to derive new solutions.	A case-base maintenance module can add, edit, or prune cases according to case usage. Integrates two design perspectives and supports the retrieval of contextually similar cases. This eases adaptation. Has identified difficulties in formulating case content and the determination of weights for case features.

The comparison of CBD systems has shown how the field originated with the development of CYCLOPS and STRUPLE as precedent-based and experience-based systems, respectively. We have analyzed how each of the systems has addressed the issues of case representation, retrieval, presentation, adaptation, and maintenance. It is evident that case representation and retrieval have been the issues dealt with by most systems. Case presentation and adaptation issues were considered by a considerable number of systems. Case-base maintenance can be seen as the issue least dealt with. Ethical and legal issues of case ownership and reuse have not been dealt with at all, although these issues will be of importance if CBD systems are to be used commercially.

Table 3 provides a summary of the case-based design systems discussed in Section 6.

8. CONCLUSION

Case-based reasoning, as defined by Riesbeck and Schank (1989), is "*the process of solving new problems by adapting solutions that were used to solve old problems.*" Case-based reasoning encourages the reuse of solutions instead of solving problems from scratch. The usefulness of this technique is enhanced when the problem to be solved is of a complex nature and where problem-solving methods are not well understood. Design is often complex and ill-structured with no generally accepted theory. Moreover, there is rarely one correct design solution, but often many possible solutions. This makes model-based solutions dependent on a strong domain theory inadequate for design problem solving. Consequently, CBR, which supports problem solving in unstructured domains, is well suited to design.

An analysis of the design process showed that the Propose–Critique–Modify design methodology matches the CBR cycle of retrieve–reuse–revise. This again supports the conclusion that CBR is an ideal technique for solving design problems. This further is supported by the fact that architects reuse parts of previous designs in developing new designs.

Building designs are highly contextual and components or elements of solutions are interdependent and constrained. In such situations adaptation of design solutions requires extensive domain knowledge. Therefore, CBD systems for building design are usually hybrid systems with case-based and knowledge-based components.

This paper identified social issues such as creativity and the ownership of designs as important. The CBD systems reviewed, however, concentrate on technical issues of case representation, retrieval, presentation, and adaptation. Several systems reviewed do, however, illustrate the importance of usability and CBD systems developed in the future will need to address social issues in greater detail.

The success of many academic CBD systems proves the validity of the concept of CBD, but a true measure of success will only be available when CBD systems are used commercially. Perhaps the most valuable contribution of CBD is its ability to divide complex problems into reusable fragments that can then be synthesized into a coherent solution. This is of generic importance to other complex problems such as scheduling and planning.

Future research in CBD needs to address many issues before it could be accepted commercially. These issues include the following:

1. Human Computer Interaction (HCI) needs further examination. This mainly relates to case presentation. The use of multimedia as a technique for improving HCI is a useful avenue to pursue.
2. The management of case bases needs further research. The design worthiness of cases must be evaluated prior to their retention as a new case. Design cases may be analyzed using feedback from implemented designs. Research in this direction could improve the quality of design cases and thereby the quality of design solutions.
3. In areas where design knowledge is owned by different people, as in building, different views of a design case need to be supported. This can provide a platform for a more collaborative design process.
4. Further research is required on hybrid architectures that integrate other AI techniques. This will be vital to support case adaptation.
5. CBD systems need effective validation and verification methods.

In conclusion, case-based design is a challenging problem. There have been many worthwhile implementations, each using different methods. It is perhaps possible now to identify a common approach for the future. Design cases are decomposed using domain knowledge and are represented hierarchically, probably using object-oriented techniques. Cases include design data and can be presented as plans, along with other pertinent information. Retrieval of past designs uses contextual information, and new designs can be adapted and composed from parts of previous designs. Adaptation and case combination are not fully automated—designers are included in the loop and any adaptations they make can be checked using verification knowledge and previous design cases.

The next step is to leave the laboratory and face a whole new set of problems in applying these techniques commercially.

ACKNOWLEDGMENTS

The authors acknowledge the Association of Commonwealth Universities, the University of Moratuwa, Sri Lanka and EPSRC whose grants (Nos. GR/J42496 & GR/L16330) helped us complete this work.

REFERENCES

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations and system approaches. *Europ. J. Artif. Intell. AICOM 7 (1)*, 39–59.

Akin, O. (1986). *Psychology of architectural design*. Pion, London.

Akin, O. (1988). Expertise of the architect. In *Expert Systems for Engineering Design* (Rychener, M., Ed.), pp. 173–196. Academic Press, London.

Alshawi, M. (1995). Dynamic generation of design plans at the brief stage. CAAD Futures '95, NUS, Singapore.

Arciszewski, T., & Michalski, R.S. (1994). Inferential design theory: A conceptual outline. In *Artificial Intelligence in Design '94* (Gero, J.S., & Sudweeks, F., Eds.), pp. 295–308. Kluwer Academic Publishers, The Netherlands.

Asimov, M. (1965). *Introduction to design*. Prentice Hall, New York.

Bakhtari, S., & Bartsch-Sporl, B. (1994). Bridging the gap between AI technology and design requirements. In *Artificial Intelligence in Design* (Gero, J.S., & Sudweeks, F., Eds.), pp. 753–768. Kluwer Academic Publishers, The Netherlands.

Barber, J., Bhatta, S., Goel, A., Jacobsen, M., Pearce, M., Penberthy, L., Shankar, M., & Stroulia, E. (1992). AskJef: Integrating case-based reasoning and multimedia technologies for interface design support. In *Artificial Intelligence in Design '92* (Gero, J.S., Ed.). Kluwer Academic Publishers, Boston, MA.

Bardasz, B., & Zeid, I. (1993). Dejavu: Case-based reasoning for mechanical design. *AIEDAM 7 (2)*, 111–124.

Barletta, R., & Mark, W. (1988). Explanation-based indexing of cases. *Proc. 7th National Conf. Artif. Intell.*

Bartsch-Sporl, B. (1994). Representing and using cases in visualisation-oriented design domains. In *Proc. 2nd EWCBR '94* (Kean, M., Haton, J.P., & Manago, M., Eds.), pp. 253–261. Springer, Berlin.

Bartsch-Sporl, B. (1995). Towards the Integration of Case-Based, Schema-Based and Model-Based Reasoning for Supporting Complex Design Tasks. In *Case-Based Reasoning Research & Development—ICCBR '95* (Veloso, M., & Aamodt, A., Eds.), pp. 145–156. Springer, Sesimbra, Portugal.

Brock, B. (1992). *Die Schonheit in der Vernunft*. Lecture at ETH on May 12, Zurich.

Carbonell, J.G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In *Machine Learning—An Artificial Intelligence Approach, Vol. 2*, pp. 371–391. Morgan Kaufmann, San Mateo, CA.

Chandrasekaran, B. (1990). Design problem solving: A task analysis. *AAAI, AI Magazine*, 59–71.

Communications of the ACM. (1995). December 38 (12).

Coulon, C.H., & Steffens, R. (1994). Comparing fragments by their images. In *Similarity Concepts and Retrieval Methods* (Vob, A., Ed.), pp. 36–44. GMD, Sankt Augustin.

Coulon, C.H. (1995). *Automatic Indexing Retrieval and Reuse of Topologies in Architectural Layouts*. CAAD Features 1995, NUS, Singapore.

Darke, J. (1984). *Developments in design methodology*. Wiley, Chichester.

Dixon, J.R. (1988). On research methodology towards a scientific theory of engineering design. *AIEDAM 1 (3)*.

Dixon, J.R., & Simmons, M.K. (1983). Computers that design: Expert systems for mechanical engineers. *Comput. Mech. Eng. 2 (3)*, 10–18.

Dixon, J.R., & Simmons, M.K. (1984). An architecture for the application of artificial intelligence to design. *Proc. ACM/IEEE—21st Annual Design Automation Conf.*, pp. 634–640.

Domeshek, E., & Kolodner, J. (1991). Toward a case-based aid for conceptual design. *Int. J. Expert Syst. 4 (2)*, 201–220.

Domeshek, E., & Kolodner, J. (1992). A case-based design aid for architecture. *Artificial Intelligence in Design '92*, pp. 497–516. The Netherlands.

Domeshek, E., & Kolodner, J. (1993). Using the points of large cases. *AIEDAM 7 (2)*, 87–96.

Domeshek, E.A., Zimring, C.M., & Kolodner, J.L. (1994). *Scaling up is hard to do: Experiences in preparing a case-based design aid prototype for field trial, ASCE*. American Society of Civil Engineers '94, pp. 430–437.

FABEL (1993*a*). Report No. 2, Survey of FABEL, The FABEL Consortium.

FABEL (1993*b*). Report No. 13. In *Similarity Concepts and Retrieval Methods* (Vob, A., Ed.).

FABEL (1994). Report No. 24, Evaluation of Retrieval Methods in Case-Based Design, Carl-Helmut Coulon. Friedrich Cebhardt.

FABEL (1996). http://nathan.gmd.de/projects/fabel.html. World-Wide Web.

Faltings, B. (1991). Case-based representation of architectual design knowledge. In *Computational Intelligence 3* (Cercone, N., Ed.), pp. 273–281. Amsterdam, Holland.

Fey, V., & Vertkin, I. (1993). The theory of inventive problem solving for systems engineering and logistics support. *Proc. Int. Symposium Advances in Logistics*, Exeter University, Exeter, UK.

Flemming, U. (1994). Case-based design in the SEED system. *Automation in Construction 3*, 123–133.

Flemming, U., Baykan, C.A., Coyne, R.F., & Fox, M.S. (1992). *Hierarchical generate-and-test vs. constraint-directed search*. Engineering Design Research Centre, Carnegie Mellon University, Pittsburgh, PA.

Flemming, U., Coyne, R., & Woodbury, R. (1993). SEED: A software environment to support the early phases in bulding desing. *ARECDAO '93 Proc. Fourth Int. Conf. on CAD in Architecture and Civil Eng.*, pp. 111–122.

Flemming, U., Coyne, R., & Snyder, J. (1994). Case-based design in the SEED system. *ASCE*, pp. 446–453.

Gentner, D., & Forbus, K.D. (1991). MAC/FAC: A model of similarity-based retrieval. *Proc. 13th Conf. Cognitive Science*, pp. 504–509.

Gero, J.S. (1987). Expert systems in computer aided design. *Proc. IFIP W5.2 Working Conf. on Expert Systems in Computer Aided Design*.

Gero, J.S. (1990). Design prototypes: A knowledge representation schema for design. *AI Magazine II (4)*, 26–36.

Gero, J.S., & Maher, M.L. (1992). Mutation and analogy to support creativity in computer-aided design. *CAAD Futures '91*, pp. 261–270. Vieweg, Wiesbaden.

Goel, A. (1989). *Integration of case-based reasoning and model-based reasoning for adaptive design problem solving*. Ph.D. Thesis, Department of Computer & Information Science, The Ohio State University, Columbus, OH.

Goel, A., Kolodner, J.L., Pearce, M., Billington, R., & Zimring, C. (1991). Towards a case-based tool for aiding conceptual design problem solving. *Proc.: Workshop on Case-based Reasoning (DARPA)*. Morgan Kaufmann, Washington, D.C.

Goldberg, D.E. (1989). *Genetic algorithms in search optimisation and machine learning*. Addison-Wesley, Reading, MA.

Grather, W. (1994). Computing distance between attribute-value representations in an associative memory. In *Similarity Concepts and Retrieval Methods* (Vob, A., Ed.), pp. 12–25. GMD, Sankt Augustin.

Haller, F. (1974). *MIDI—ein offenes system fur mehregeschossige bauten mit integrierter medieninstallation*. USM baausysteme haller, Munsingen.

Haller, F. (1988). *bauen und forschen*. Fritz Haller Solothurn, Ausstellung des Knustvereins Solothurn.

Hanney, K., Keane, M., Smyth, B., & Cunningham, P. (1995). What kind of adaptation do CBR system sneed?: A review of current practice. Review of Adaptation in CBR Systems, *Proc. AAAI Fall Symposium Technical Report FS-95-02*. AAAI Press.

Hayes, J.R. (1981). *The complete problem solver*. The Franklin Institute Press, Philadelphia, PA.

Hennessy, D., & Hinkle, D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert 7 (5)*, 21–26.

Hinrichs, T.R. (1988). Towards an architecture for open world problem solving. *Proc. Workshop on CBR (DARPA)*. Morgan Kaufmann, New York.

Hua, K., & Faltings, B. (1993). Exploring case-based building design-Cadre. *AIEDAM 7 (2)*, 135–143.

Hua, K. Faltings, B., Smith, I., Schmitt, G., & Shih, S.G. (1992). Adaptation of spatial design cases. *2nd International Conference on AI in Design*, pp. 559–575. Kluwer Academic Publishers, The Netherlands.

Huff, C., & Martin, C.D. (1995). Computing consequences: A framework for teaching ethical computing. *Communications of the ACM 38 (12)*, 75–84.

Hunt, J. (1995). Evolutionary case based design. In *Progress in Case-Based Reasoning* (Watson, I.D., Ed.), pp. 17–31. Springer, Salford, UK.

Hunt, J., & Miles, R. (1995). Towards an intelligent architectural design aid. *Expert Systems 12 (3)*, 209–218.

Keeney, R., & Raiffa, H. (1976). *Decisions with multiple objectives: Preferences and value tradeoffs*. Wiley, New York.

Kolodner, J.L. (1988). *Proceedings of the 1st case-based reasoning workshop*. Morgan Kaufmann, New York.

Kolodner, J.L. (1993). *Case-based reasoning*. Morgan Kaufmann, New York.

Koton, P. (1989). *Using experience in learning and problem solving*. Ph.D. Thesis, MIT/LCS/TR-441, Laboratory of Computer Science.

Lehane, M.S., & Moore, C.J. (1996). Applying case-based reasoning in bridge design. In *Information Technology in Civil and Structural Engineering Design* (Kumar, B., MacLeod, I.A., & Retik, A., Eds.), pp. 1–6. Civil-Comp Limited, UK.

Linowski, B. (1994). Computing distances between attribute-value representations in a flat memory. In *Similarity Concepts and Retrieval Methods* (Vob, A., Ed.), pp. 26–35. GMD, Sankt Augustin.

Mackinder, M., & Marvin, H.E. (1982). *Design decision making in architectural practice*. Research paper No. 19, Institute of Advanced Architectural Studies, University of York.

Maher, M.L. (1987). Engineering design synthesis: A domain independent representation. *AIEDAM 1 (3)*, 207–213.

Maher, M.L. (1994*a*). Representation of case memory for structural design. *ASCE* 2030–2037.

Maher, M.L. (1994*b*). Creative design using a genetic algorithm. *ASCE* 2014–2021.

Maher, M.L., & Balachandran, B. (1994*a*). Flexible retrieval strategies for case-based design. In *Artificial Intelligence in Design '94* (Gero, J.S., & Sudweeks, F., Eds.), pp. 163–180. Kluwer Academic Publishers, The Netherlands.

Maher, M.L., & Balachandran, B. (1994*b*). Multimedia approach to case-based structural design. *J. Comput. Civil Eng. 8 (3)*, 359–376.

Maher, M.L., & Zhang, D.M. (1991). CADSYN: Using case and decomposition knowledge for design synthesis. In *Artificial Intelligence in Design* (Gero, J., Ed.).

Maher, M.L., & Zhang, D.M. (1993). CADSYN: A case-based design process model. *AIEDAM 7 (2)*, 97–110.

Maher, M.L., & Zhao, F. (1987). Using experience to plan the synthesis of new designs. In *Expert Systems in Computer Aided Design* (Gero, J.S., Ed.), pp. 349–373. Elsevier Science Publishers B.V., The Netherlands.

Maher, M.L., Balachandran, M.B., & Zhang, D.M. (1995). *Case-based reasoning in design*. Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ.

Marir, F., & Watson, I.D. (1995*a*). CBRefurb: A case-based building refurbishment cost estimator and decision support system. In *Developments in Artificial Intelligence for Civil and Structural Engineering* (Topping, B.H.V., Ed.), pp. 231–236. Civil-Comp Press, Edinburgh, UK.

Marir, F., & Watson, I.D. (1995*b*). Representing and indexing building refurbishment cases for multiple retrieval of adaptable pieces of cases. In *Case-Based Reasoning Research & Development—ICCBR '95* (Veloso, M., & Aamodt, A., Eds.), pp. 55–66. Springer, Sesimbra, Portugal.

Mittal, S., Dym, C.L., & Morjaria, M. (1985). Pride: An expert system for the design of paper handling systems. In *Application of Knowledge Based Systems*, pp. 99–115. American Society of Mechanical Engineers.

Moore, C.J., & Lehane, M.S. (1996). A case-base for decision support in bridge design. *IEEE Expert*.

Mostow, J. (1989). Design by derivational analogy: Issues in the automated replay of design plans. *Artif. Intell.* 119–184.

Navinchandra, D.J. (1987). *Exploring for innovative designs by relaxing criteria and reasoning from precedent knowledge*. D.Sc. Thesis, Massachusetts Institute of Technology.

Navinchandra, D. (1988). Case-based reasoning in CYCLOPS, a design problem solver. *Proceedings: Workshop on Case-Based Reasoning (DARPA)*, pp. 286–301. Morgan Kaufmann, Clearwater, FL, San Mateo, CA.

Navinchandra, D. (1991). *Exploration and innovation in design: Towards a computational model*. Springer-Verlag, New York.

Navinchandra, D., Sycara, K.P., & Narasimhan, S. (1991). Behavioural synthesis in CADET, a case-based design tool. *Proc. Seventh IEEE Conf. on AI Applications*, pp. 217–221. IEEE Press, Miami, New York.

Newell, A. (1980). Reasoning, problem solving and decision process: The problem space as a fundamental category. *Attention and Performance 8*, 693–718.

Oxman, R.E. (1993). INDEX: A case-based reasoning approach of content-based indexing for design. *The 8th Int. Conf. Application of Artif. Intell. Vol. 1*, pp. 201–218.

Oxman, R. (1994). *Precedents in design: A computational model for the organisation of case knowledge*, pp. 438–445. American Society of Civil Engineers '94.

Oxman, R. (1995). Design case bases: Graphic knowledge bases for the design workspace. *Int. Conf. Computer Aided Architectural Design—CAAD*. Futures, Singapore.

Oxman, R., & Oxman, R. (1993*a*). Remembrance of things pase: Design precedents in libraries. *Automation in Construction 2*, 21–29.

Oxman, R., & Oxman, R. (1993*b*). PRECEDENTS: Memory structure in design case libraries. In *CAAD Futures '93* (Flemming, U., & Wyk, S.V., Eds.), pp. 273–287. Elsevier Science Publishers B.V., The Netherlands.

Pahl, G., & Beitz, W. (1984). *Engineering design*. Springer-Verlag, New York.

Pearce, M., Goel, A., Kolodner, J.L., ZImring, C. Sentosa, L., & Billington, R. (1992). Case-based design support: A case study in architectural design. *IEEE Expert 7 (5)*, 14–20.

Perera, A.A.D.A.J. (1989). *Cost effective designs*. Ph.D. Thesis, Loughborough University of Technology, UK.

Perera, R.S., Watson, I.D., & Alshawi, M. (1995). Case-based design approach for integration of design and estimating. *Proc. 4th Int. Conf. Application of AI to Civil & Structural Eng.*

Perera, R.S. & Watson, I.D. (1995). NIRMANI: An integrated case-based system for strategic design and estimating. *Proc. First UK CBR Workshop, Progress in Case-Based Reasoning*, pp. 185–200. Springer-Verlag, Salford, UK.

Perera, R.S., & Watson, I.D. (1996). Multi-agent collaborative case-based estimating and design in NIRMANI: Organising a multi-perspective case memory. In *Information Processing in Civil and Structural Engineering Design* (Kumar, B., Ed.), pp. 53–64. Civil-Comp Press, Scotland, UK.

Pu, P. (1993). Introduction: Issues in case-based design systems. *AIEDAM 7 (2)*, 79–85.

Pu, P., & Reschberger, M. (1991). Assembly sequence planning using case-based reasoning technique. In *Artificial Intelligence in Design '91*. Butterworth Heinemann, London.

Raphael, B., Kumar, B., & McLeod, I.A. (1994). Representing design cases based on methods. *ASCE* 285–292.

Riesbeck, C.K., & Schank, R.S. (1989). *Inside case-based reasoning*. Erlbaum, Northvale, NJ.

Roderman, S., & Tsatsoulis, C. (1993). PANDA: A case-based system to aid novice designers. *AIEDAM 7 (2)*, 125–133.

Rosenman, M.A., & Gero, J.S. (1993). Creativity in design using a design prototype approach. In *Modelling Creativity and Knowledge-Based Creative Design* (Gero, J.S., & Maher, M.L., Eds.), pp. 111–138. Lawrence Erlbaum Associates, Inc. Publishers, Hillsdale, NJ.

Schaaf, J.W. (1994). Using gestalten to retrieve cases. *Proc. EWCBR '94*, pp. 75–83.

Schaaf, J.W., & Voss, A. (1995). Retrieval of similar layouts in FABEL using AspecT. In *CAAD Features 1995*. NUS, Singapore.

Schank, R. (1986). Explanation patterns. In *Understanding mechanically and creatively*. Lawrence Erlbaum, Hillsdale, NJ.

Schank, R., & Abelson, R. (1975). Scripts, plans, and knowledge. *Proc. Fourth Int. Joint Conf. Artif. Intell.*, pp. 151–157.

Schank, R.C., & Abelson, R.P. (1977). *Scripts, plans, goals and understanding*. Erlbaum, Hillsdale, NJ.

Schmitt, G. (1993*a*). Case-based design and creativity. *Journal of Automation in Construction 2*, 11–19.

Schmitt, G. (1993*b*). Case-based reasoning in an integrated design and construction system. *Int. J. Construction Inf. Technol. 1 (3)*, 39–51.

Schmitt, G., Bailey, S.F., & Smith, I.F.C. (1994). Advances and challenges in case-based design. *ASCE*, 301–309.

Shih, S.G. (1994). *The use of string grammars in architectural design*. Ph.D. dissertation, Dept. of Architecture, ETH, Zurich, Switzerland.

Simon, H.A. (1973). The structure of ill structured problems. *Artif. Intell. 4*, 181–201.

Simoudis, E., Mendall, A., & Miller, P. (1993). Automated support for developing retrieve-and-propose systems. *Proc. Artif. Intell. XI Conf.*

Simpson, R.L. (1985). *A computer model of case-based reasoning in problem solving*. An investigation in the domain of dispute mediation, Atlanta, GA.

Slade, S. (1991). Case-based reasoning: A research paradigm. *AI Magazine* 42–55.

Smith, I.F.C., & Faltings, B.V. (1994). Spatial design of complex artefacts using cases. *Proc. 10th Conf. AI for Applications*, pp. 70–76.

Smith, I., Lottaz, C., & Faltings, B. (1995). Spatial composition using cases: IDIOM. *ICCBR '95*, pp. 88–97.

Smyth, B., & Keane, M. (1996). Adaptation-guided retrieval: Using adaptation knowledge to guide the retrieval of adaptable cases. In *Progress*

in Case-Based Reasoning– UKCBR2 (Watson, I.D., Ed.), pp. 2–15. Salford University, Salford, UK.

Stallman, R., & Sussman, G. (1977). Forward reasoning and dependency-directed backtracing in a system for computer-aided circuit analysis. *Artif. Intell. 9*, 135–196.

Suh, N.P. (1990). *Principles of design*. Oxford University Press, UK.

Sycara, K. (1992). A case-based synthesis tool for engineering design. *Int. J. Expert Systems 4 (2)*, 157–188.

Sycara, K.P., & Navinchandra, D. (1989). Integrating case-based reasoning and qualitative reasoning in engineering design. In *Artificial Intelligence in Engineering Design* (Gero, J.S., Ed.). Computational Mechanics Publications, UK.

Sycara, K.P., & Navinchandra, D. (1992). Retrieval strategies in a case-based design system. In *Artificial Intelligence in Engineering Design* (Tong, C., & Sriram, D., Eds.), Vol. 2, pp. 145–163. Academic Press, New York.

Sycara, K.P., Navinchandra, D., Guttal, R., Koning, J., & Narasimhan, S. (1992). CADET: A case-based synthesis tool for engineering design. *Int. J. Expert Systems 4 (2)*, 157–188.

Tanaka, T., Hattori, M., & Sueda, N. (1992). Use of multiple cases in case-based design. *Proc. IEEE Computer Soc.*, pp. 233–239.

Tomivama, T., & Yoshikawa, H. (1987). *Extended general design theory*. Design Systems for CAD, North Holland.

Watson, I.D., & Abdullah, S. (1994). Developing case-based reasoning systems: A case study in diagnosing building defects. *Proc. IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, pp. 1/1–1/3.

Watson, I.D., & Marir, F. (1994). Case-based reasoning: A review. *Knowledge Eng. Rev. 9 (4)*, 327–354.

Watson, I.E., & Perera, R.S. (1995). NIRMANI: A case-based expert system for integrated design & estimating. *Proc. Expert Systems 95*, pp. 335–348.

Zhao, F., & Maher, M.L. (1988). Using analogical reasoning to design buildings. *Eng. with Comput. 4 (3)*, 107–119.

Zimring, C., Do, E., Domeshek, E., & Kolodner, J.L. (1995). Supporting case-study use in design education: A computational case-based design aid for architecture. *Computing Civil Eng.* 1635–1642.